AWS CLOUD COMPUTING COHORT 2

CAPSTONE PROJECT 2

SERVERLESS INFRASTRUCTURE-AS-CODE (IAC) SOLUTION FOR AUTOMATED LANGUAGE TRANSLATION ON AWS

EUGENE QUAYE TETTEH

AZUBI AFRICA

5TH SEPTEMBER, 2025

Table of Contents

Table of Contents

Project Overview

This project delivers a fully serverless, Infrastructure-as-Code (IaC) solution for automated language translation on AWS, using CloudFormation or Terraform to provision resources. It leverages AWS Translate for NLP processing and Amazon S3 for object storage. Translation requests in JSON format are processed using a Python script (Boto3), which stores original and translated data in separate S3 buckets. Automation is achieved with AWS Lambda, making this solution scalable, cost-effective, and easily extensible, while remaining within AWS Free Tier limits.

Objectives

- Automate end-to-end language translation using scalable, serverless AWS services, requiring no manual infrastructure management.
- Enable Continuous Deployment: All cloud resources are provisioned, updated, and decommissioned automatically using Terraform.
- Guarantee Security: Employ robust IAM roles and policies adhering to the principle of least privilege.
- Maintain Cost-Effectiveness: Operate entirely within AWS Free Tier, using lifecycle management to minimize S3 costs.
- Create Reproducible Results: All configurations, policies, and code are tracked and can be re-deployed or destroyed in any AWS account.

System Architecture

A user uploads a set of texts (as a JSON file) for translation. Upon upload, the system immediately processes the file:

Amazon S3 receives the JSON file (source texts plus source and target language codes).

An S3 Event Notification triggers the Lambda function.

The Lambda function reads the file, translates each text using AWS Translate, and writes a new JSON with translations to an S3 response bucket.

The user retrieves the translated file from S3.

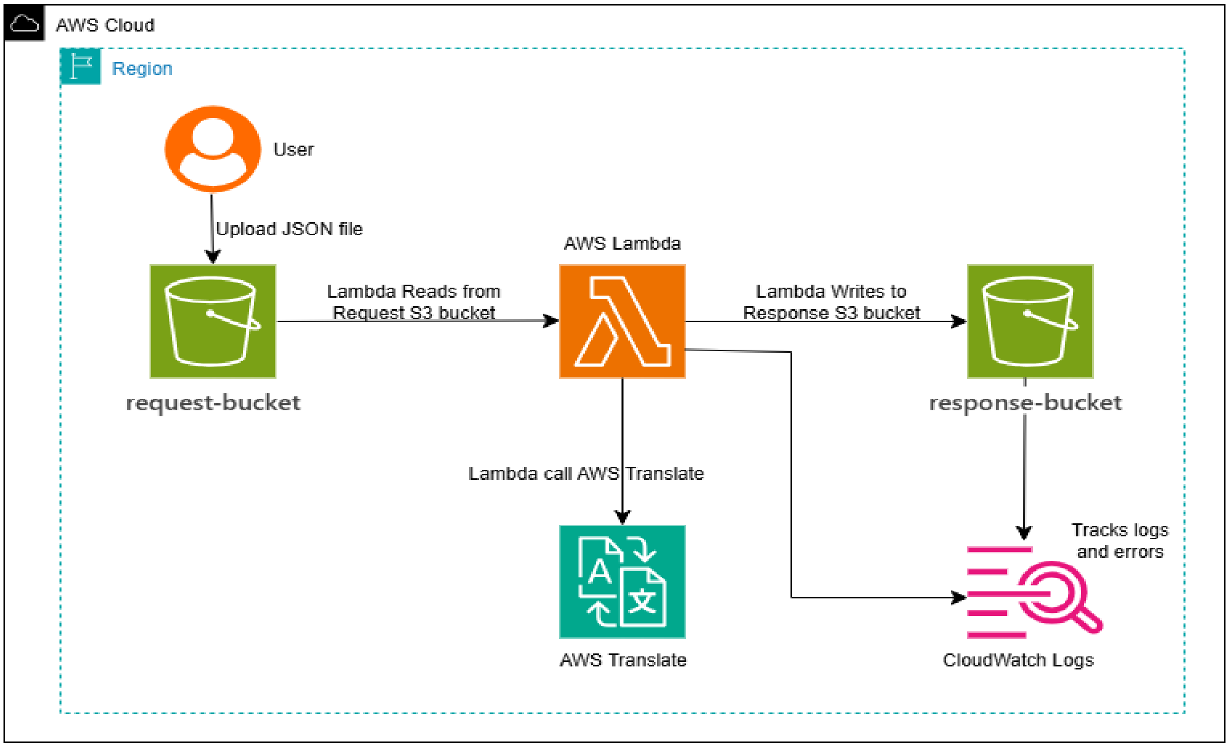
Architecture Diagram Description:

S3 (request-bucket) ← [Upload JSON]

S3 Event → Lambda (translate-json-files)

Lambda → AWS Translate API

Lambda → S3 (response-bucket) [Upload translated JSON]



Cloud Services Selection and Justification

Amazon S3

Use: Stores input files (to be translated) and output files (translated text).

Reason: Simple object storage, supports event notifications, Free Tier eligible.

AWS Lambda

Use: Serverless compute to handle all translation logic on-demand.

Reason: No server management; scales instantly with usage; 1M free executions/month.

AWS Translate

Use: Performs the actual language translation.

Reason: Pre-trained, neural machine translation, Free Tier up to 2M characters/month.

AWS IAM

Use: Restricts Lambda and S3 operations to the minimum set required.

Reason: Prevents security breaches, follows security best practices.

Terraform

Use: Defines entire system state as code — provisioning, updating, and destroying cloud resources as a unified workflow.

Reason: Reproducibility, version control, and automation, which are industry best practice for cloud projects.

Prerequisites

- AWS Account (Free Tier eligible)
- AWS CLI and credentials configured (aws configure)
- Python 3.7+ installed locally
- Terraform v1.x installed
- Boto3 Python package
- Basic knowledge of AWS IAM, Lambda, S3

Infrastructure Implementation with Terraform

S3 Buckets

Two buckets: request-bucket (input) and response-bucket (output).

Both have lifecycle rules to auto-delete objects after 7 days, ensuring no unexpected charges.

IAM Policies & Roles

One role for Lambda, with a custom inline policy granting only S3 and Translate API access.

S3 buckets are locked down with only this role having access.

Lambda Function

Deployed using Terraform with all environment variables configured.

Source code packaged as a .zip and referenced by the Lambda resource.

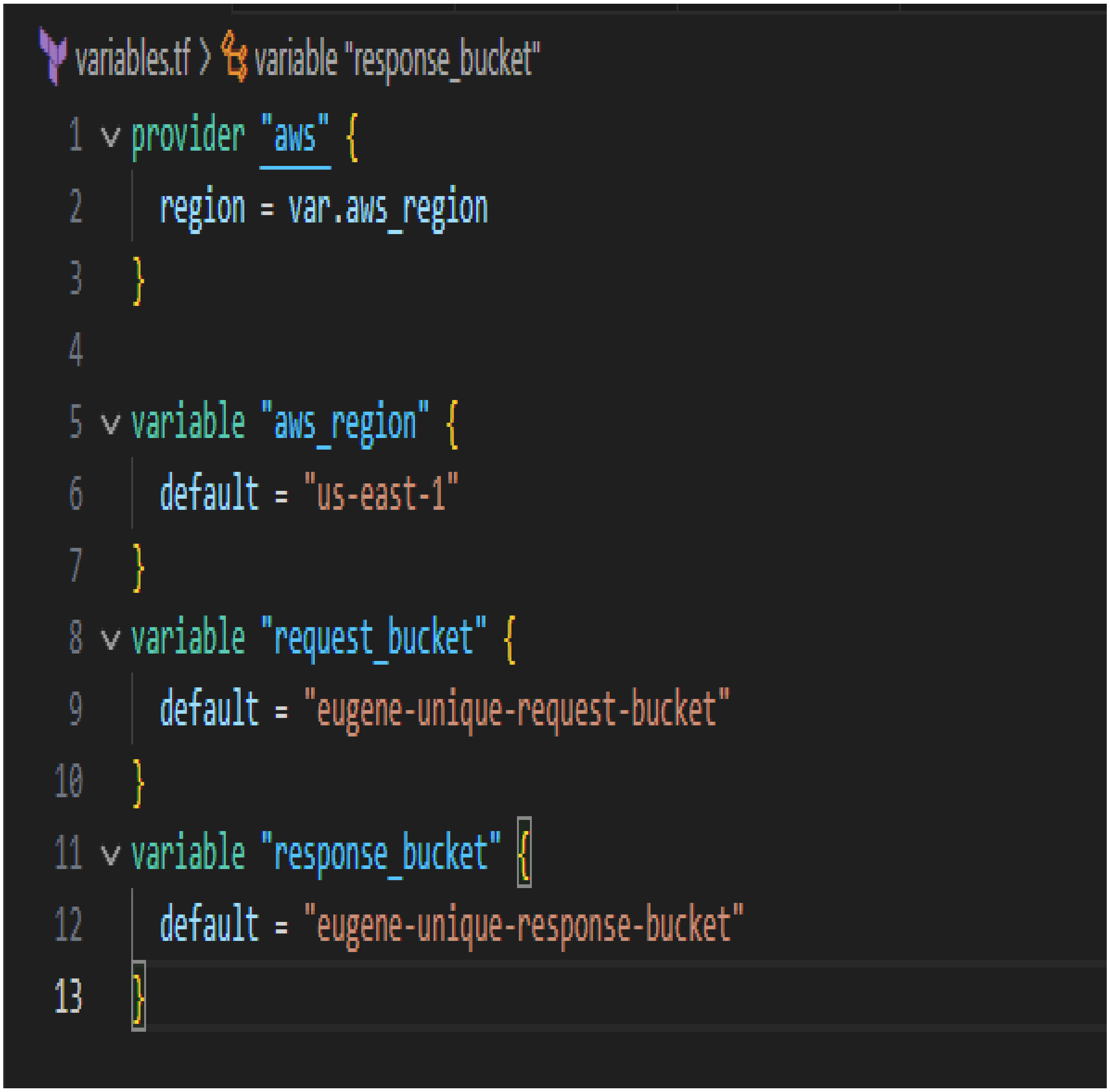
S3 Bucket Notification

Provisions an S3 event on object creation to invoke Lambda.

Ensures Lambda is only triggered when new JSON files are uploaded.

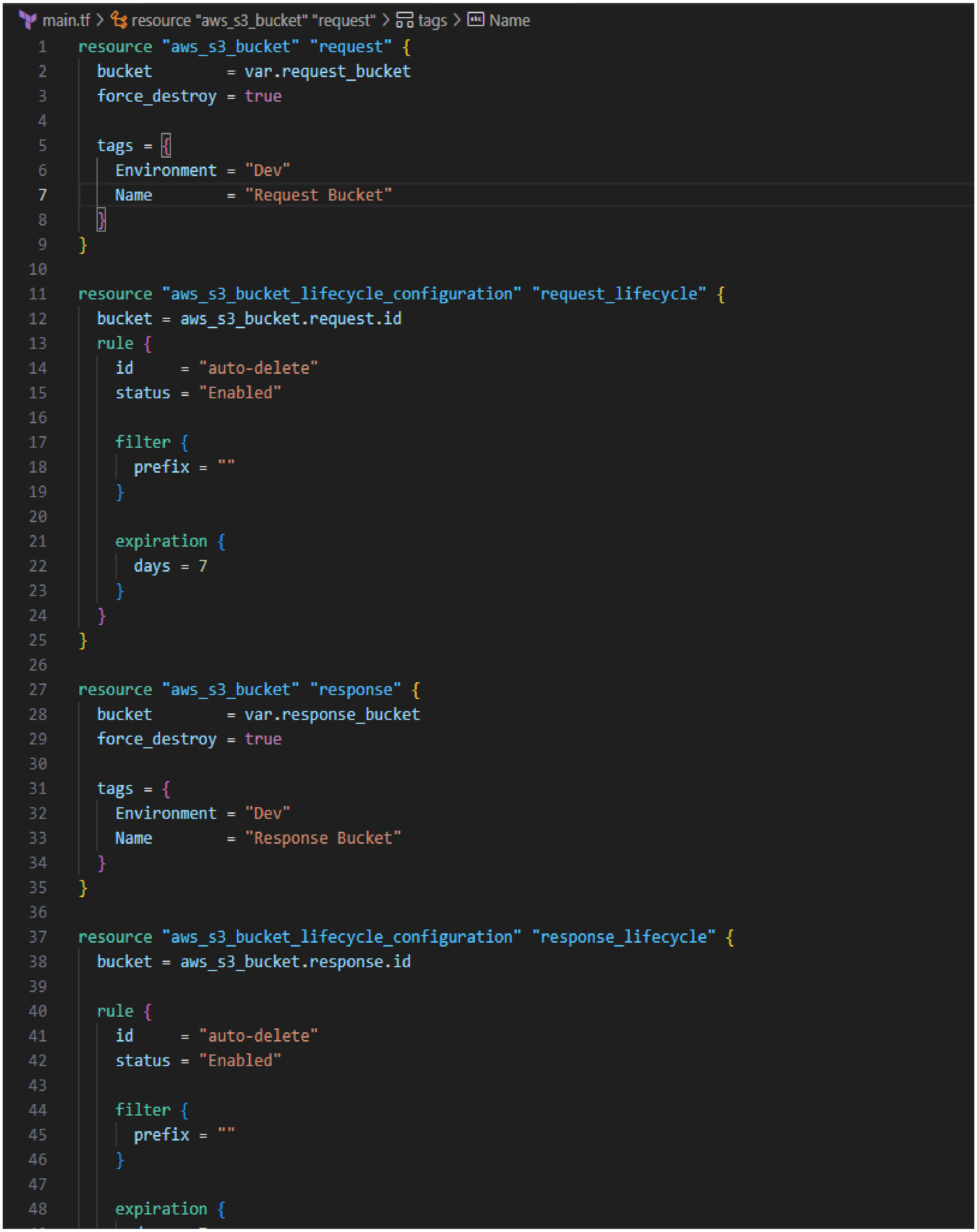
Below are the key Terraform resources:

Providers and Variables (variables.tf)



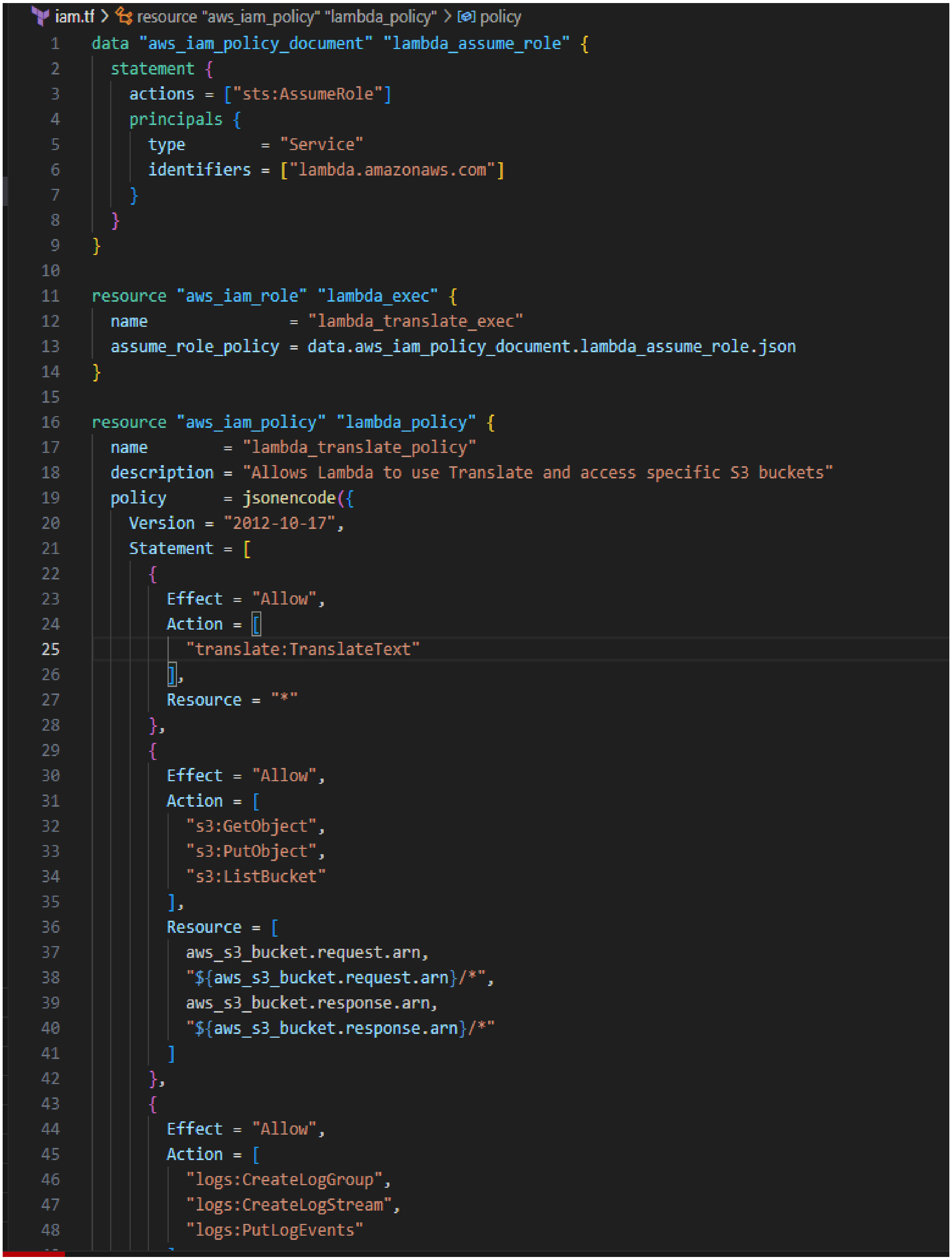
```
provider "aws" {
  region = var.aws_region
}

variable "aws_region" {
  default = "us-east-1"
}
variable "request_bucket" {
  default = "eugene-unique-request-bucket"
}
variable "response_bucket" {
  default = "eugene-unique-response-bucket"
}
```

S3 Buckets with Lifecycle Policies (main.tf)



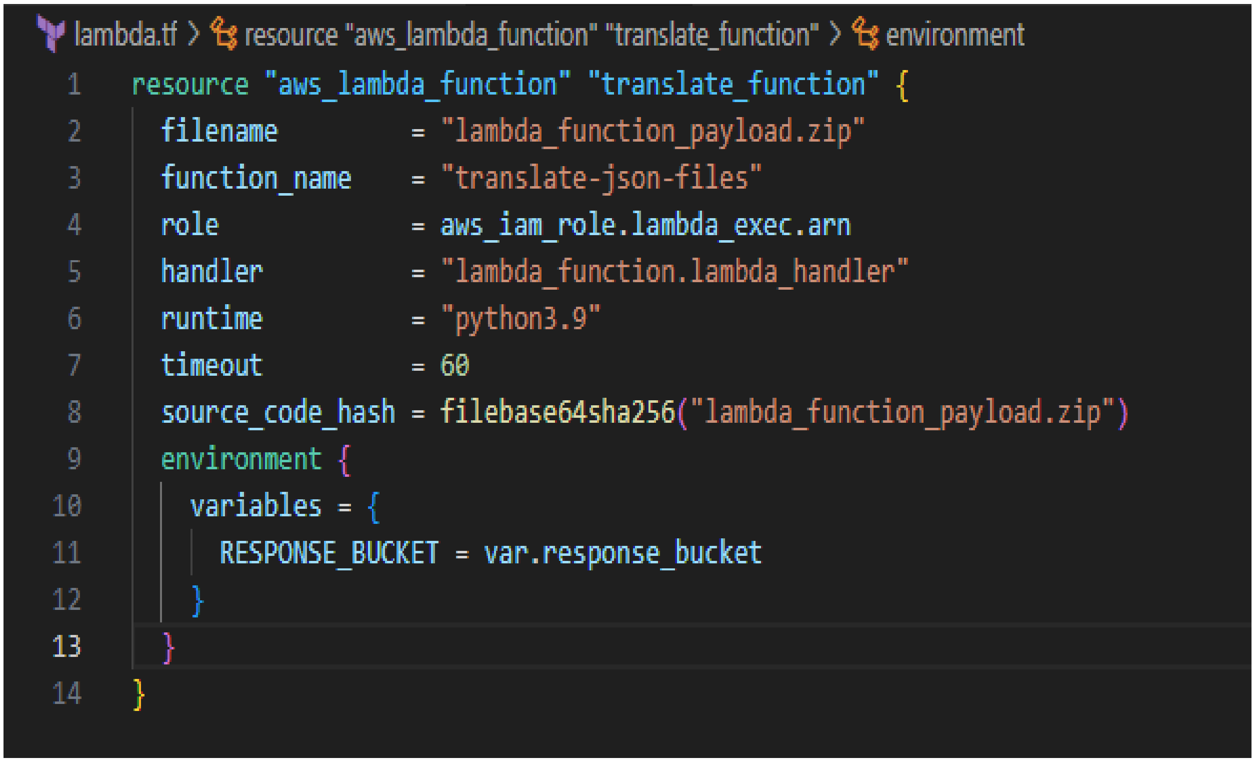
```
resource "aws_s3_bucket" "request" {
  bucket        = var.request_bucket
  force_destroy = true

  tags = {
    Environment = "Dev"
    Name        = "Request Bucket"
  }
}

resource "aws_s3_bucket_lifecycle_configuration" "request_lifecycle" {
  bucket = aws_s3_bucket.request.id
  rule {
    id     = "auto-delete"
    status = "Enabled"

    filter {
      prefix = ""
    }

    expiration {
      days = 7
    }
  }
}

resource "aws_s3_bucket" "response" {
  bucket        = var.response_bucket
  force_destroy = true

  tags = {
    Environment = "Dev"
    Name        = "Response Bucket"
  }
}

resource "aws_s3_bucket_lifecycle_configuration" "response_lifecycle" {
  bucket = aws_s3_bucket.response.id

  rule {
    id     = "auto-delete"
    status = "Enabled"

    filter {
      prefix = ""
    }

    expiration {
```

Lambda IAM Role and Policy (iam.tf)



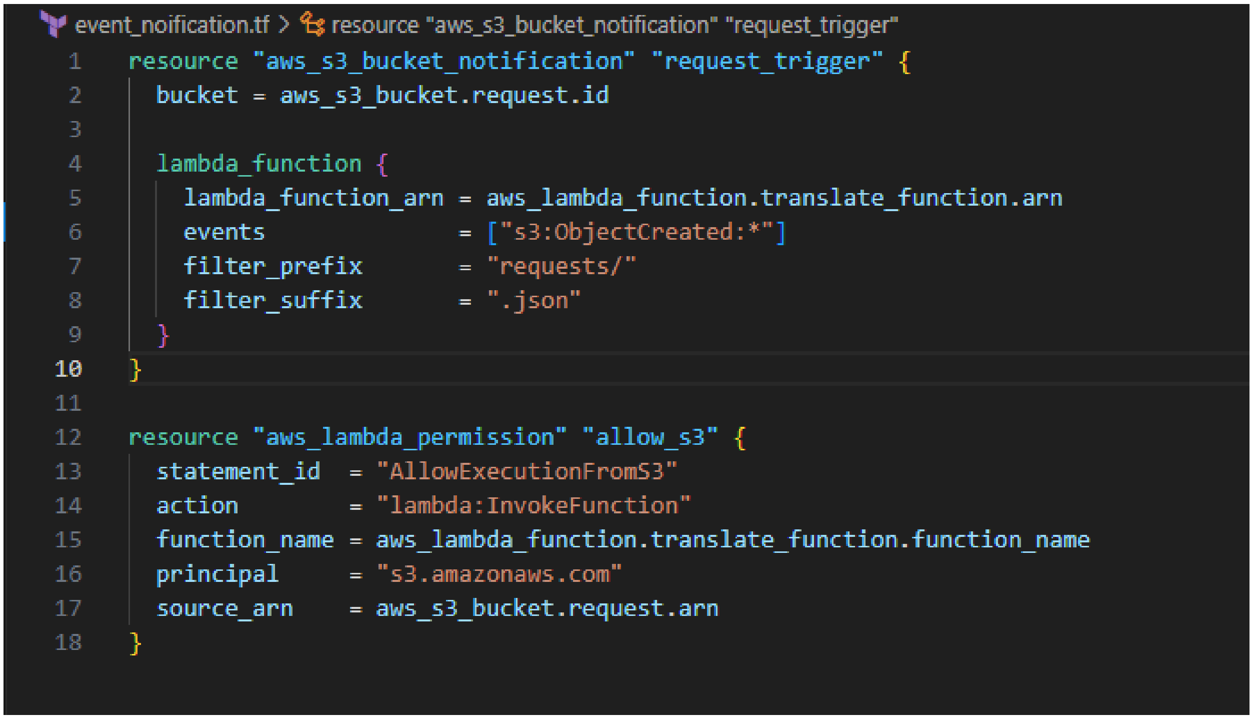
```
data "aws_iam_policy_document" "lambda_assume_role" {
  statement {
    actions = ["sts:AssumeRole"]
    principals {
      type        = "Service"
      identifiers = ["lambda.amazonaws.com"]
    }
  }
}

resource "aws_iam_role" "lambda_exec" {
  name               = "lambda_translate_exec"
  assume_role_policy = data.aws_iam_policy_document.lambda_assume_role.json
}

resource "aws_iam_policy" "lambda_policy" {
  name        = "lambda_translate_policy"
  description = "Allows Lambda to use Translate and access specific S3 buckets"
  policy      = jsonencode({
    Version = "2012-10-17",
    Statement = [
      {
        Effect = "Allow",
        Action = [
          "translate:TranslateText"
        ],
        Resource = "*"
      },
      {
        Effect = "Allow",
        Action = [
          "s3:GetObject",
          "s3:PutObject",
          "s3:ListBucket"
        ],
        Resource = [
          aws_s3_bucket.request.arn,
          "${aws_s3_bucket.request.arn}/*",
          aws_s3_bucket.response.arn,
          "${aws_s3_bucket.response.arn}/*"
        ]
      },
      {
        Effect = "Allow",
        Action = [
          "logs:CreateLogGroup",
          "logs:CreateLogStream",
          "logs:PutLogEvents"
```

Lambda Function (lambda.tf)


lambda.tf > resource "aws_lambda_function" "translate_function" > environment

```
resource "aws_lambda_function" "translate_function" {
  filename         = "lambda_function_payload.zip"
  function_name    = "translate-json-files"
  role             = aws_iam_role.lambda_exec.arn
  handler          = "lambda_function.lambda_handler"
  runtime          = "python3.9"
  timeout          = 60
  source_code_hash = filebase64sha256("lambda_function_payload.zip")
  environment {
    variables = {
      RESPONSE_BUCKET = var.response_bucket
    }
  }
}
```

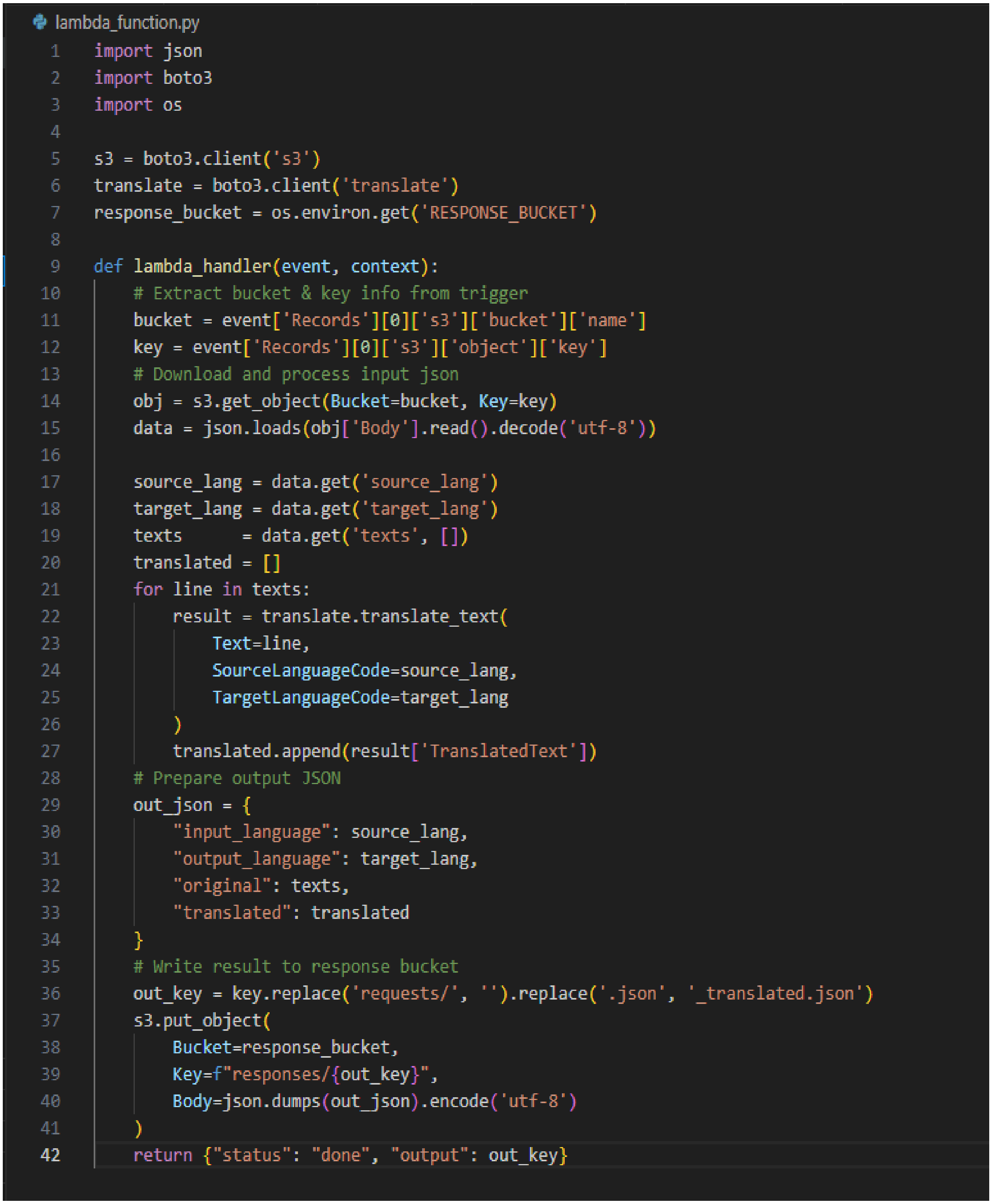
S3 Event Trigger (event_notification.tf)


event_noification.tf > resource "aws_s3_bucket_notification" "request_trigger"

```
resource "aws_s3_bucket_notification" "request_trigger" {
  bucket = aws_s3_bucket.request.id

  lambda_function {
    lambda_function_arn = aws_lambda_function.translate_function.arn
    events              = ["s3:ObjectCreated:*"]
    filter_prefix       = "requests/"
    filter_suffix       = ".json"
  }
}

resource "aws_lambda_permission" "allow_s3" {
  statement_id  = "AllowExecutionFromS3"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.translate_function.function_name
  principal     = "s3.amazonaws.com"
  source_arn    = aws_s3_bucket.request.arn
}
```

Lambda Function Development & Packaging

Python Lambda Function (lambda_function.py)



lambda_function.py

```python
import json
import boto3
import os

s3 = boto3.client('s3')
translate = boto3.client('translate')
response_bucket = os.environ.get('RESPONSE_BUCKET')

def lambda_handler(event, context):
    # Extract bucket & key info from trigger
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']
    # Download and process input json
    obj = s3.get_object(Bucket=bucket, Key=key)
    data = json.loads(obj['Body'].read().decode('utf-8'))

    source_lang = data.get('source_lang')
    target_lang = data.get('target_lang')
    texts      = data.get('texts', [])
    translated = []
    for line in texts:
        result = translate.translate_text(
            Text=line,
            SourceLanguageCode=source_lang,
            TargetLanguageCode=target_lang
        )
        translated.append(result['TranslatedText'])
    # Prepare output JSON
    out_json = {
        "input_language": source_lang,
        "output_language": target_lang,
        "original": texts,
        "translated": translated
    }
    # Write result to response bucket
    out_key = key.replace('requests/', '').replace('.json', '_translated.json')
    s3.put_object(
        Bucket=response_bucket,
        Key=f"responses/{out_key}",
        Body=json.dumps(out_json).encode('utf-8')
    )
    return {"status": "done", "output": out_key}
```

You must upload the packaged ***lambda_function_payload.zip*** and reference it locally**.**

Packaging Instructions:

1. Place the Python script as ***lambda_function.py*** in a folder
2. cd to the folder and run: ***“Compress-Archive -Path lambda_function.py -DestinationPath "C:\Users\quaye\Desktop\Capstone Project 2\lambda_function_payload.zip”***
3. Make sure you update the Lambda resource in Terraform with the correct zip path (***filename***).

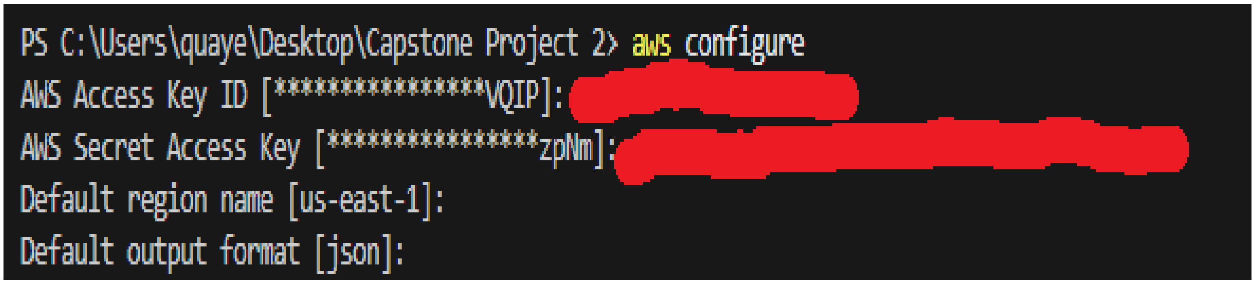
End-to-End AWS Deployment and Teardown Using Terraform

AWS CLI Configuration

Initiated the AWS CLI setup using the command: ***aws configure***

Provided the required credentials:

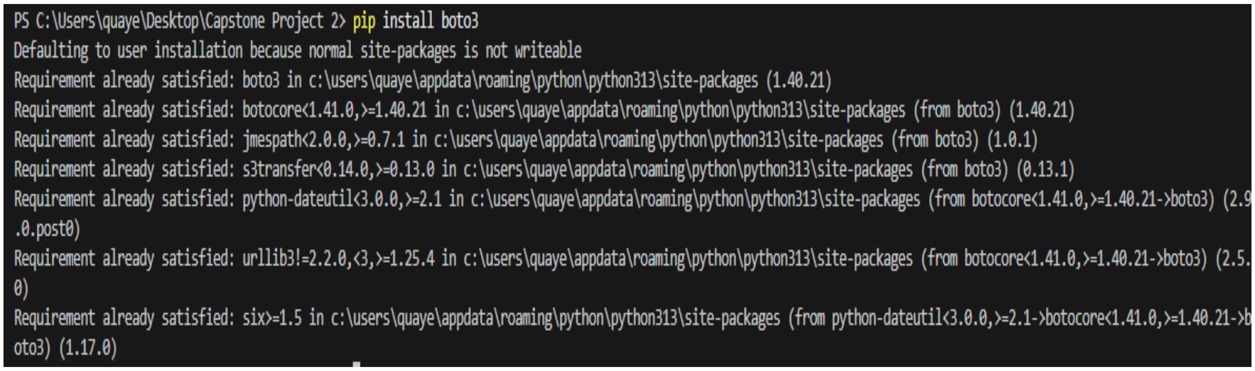
- Access Key ID
- Secret Access Key
- Default Region Name
- Output Format



```
PS C:\Users\quaye\Desktop\Capstone Project 2> aws configure
AWS Access Key ID [****************VQIP]: [redacted]
AWS Secret Access Key [****************zpNm]: [redacted]
Default region name [us-east-1]:
Default output format [json]:
```

Installing Boto3

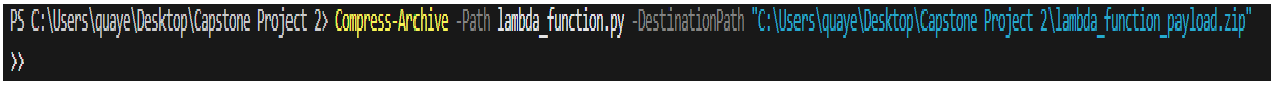
Installed the AWS SDK for Python (Boto3) to enable programmatic interaction with AWS services:



```
PS C:\Users\quaye\Desktop\Capstone Project 2> pip install boto3
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: boto3 in c:\users\quaye\appdata\roaming\python\python313\site-packages (1.40.21)
Requirement already satisfied: botocore<1.41.0,>=1.40.21 in c:\users\quaye\appdata\roaming\python\python313\site-packages (from boto3) (1.40.21)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in c:\users\quaye\appdata\roaming\python\python313\site-packages (from boto3) (1.0.1)
Requirement already satisfied: s3transfer<0.14.0,>=0.13.0 in c:\users\quaye\appdata\roaming\python\python313\site-packages (from boto3) (0.13.1)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\quaye\appdata\roaming\python\python313\site-packages (from botocore<1.41.0,>=1.40.21->boto3) (2.9
.0.post0)
Requirement already satisfied: urllib3!=2.2.0,<3,>=1.25.4 in c:\users\quaye\appdata\roaming\python\python313\site-packages (from botocore<1.41.0,>=1.40.21->boto3) (2.5.
0)
Requirement already satisfied: six>=1.5 in c:\users\quaye\appdata\roaming\python\python313\site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.41.0,>=1.40.21->b
oto3) (1.17.0)
```

Packaging Lambda Function

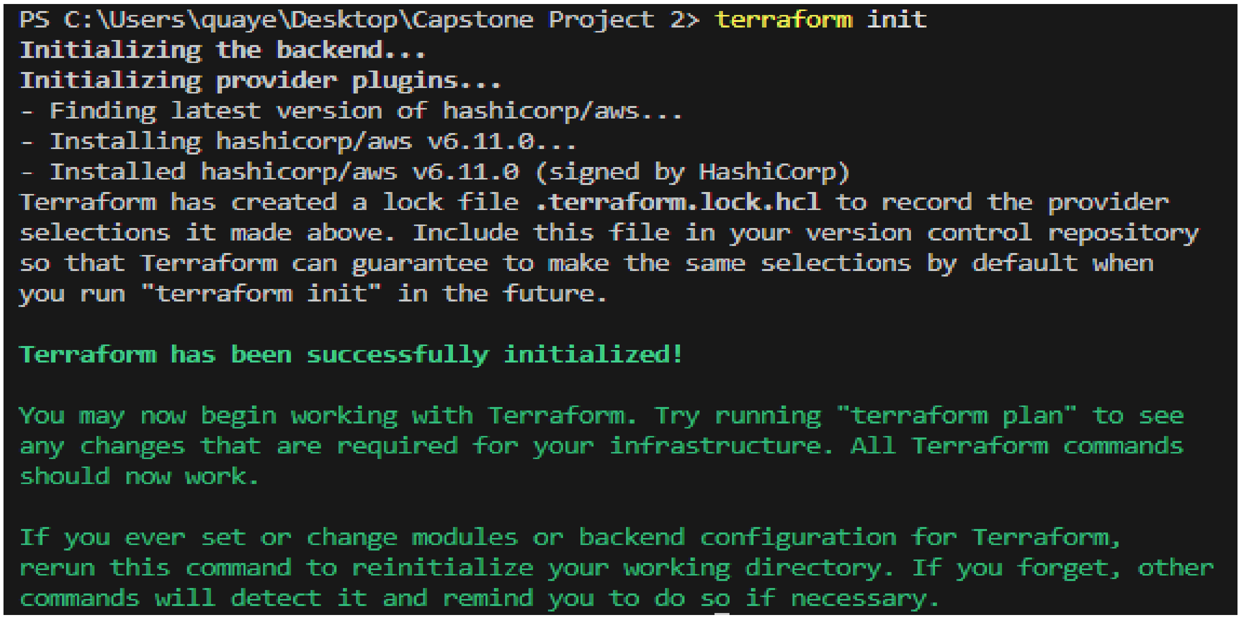
Compressed the Lambda function script into a ZIP archive for deployment:



```
PS C:\Users\quaye\Desktop\Capstone Project 2> Compress-Archive -Path lambda_function.py -DestinationPath "C:\Users\quaye\Desktop\Capstone Project 2\lambda_function_payload.zip"
>>
```

Terraform Initialized

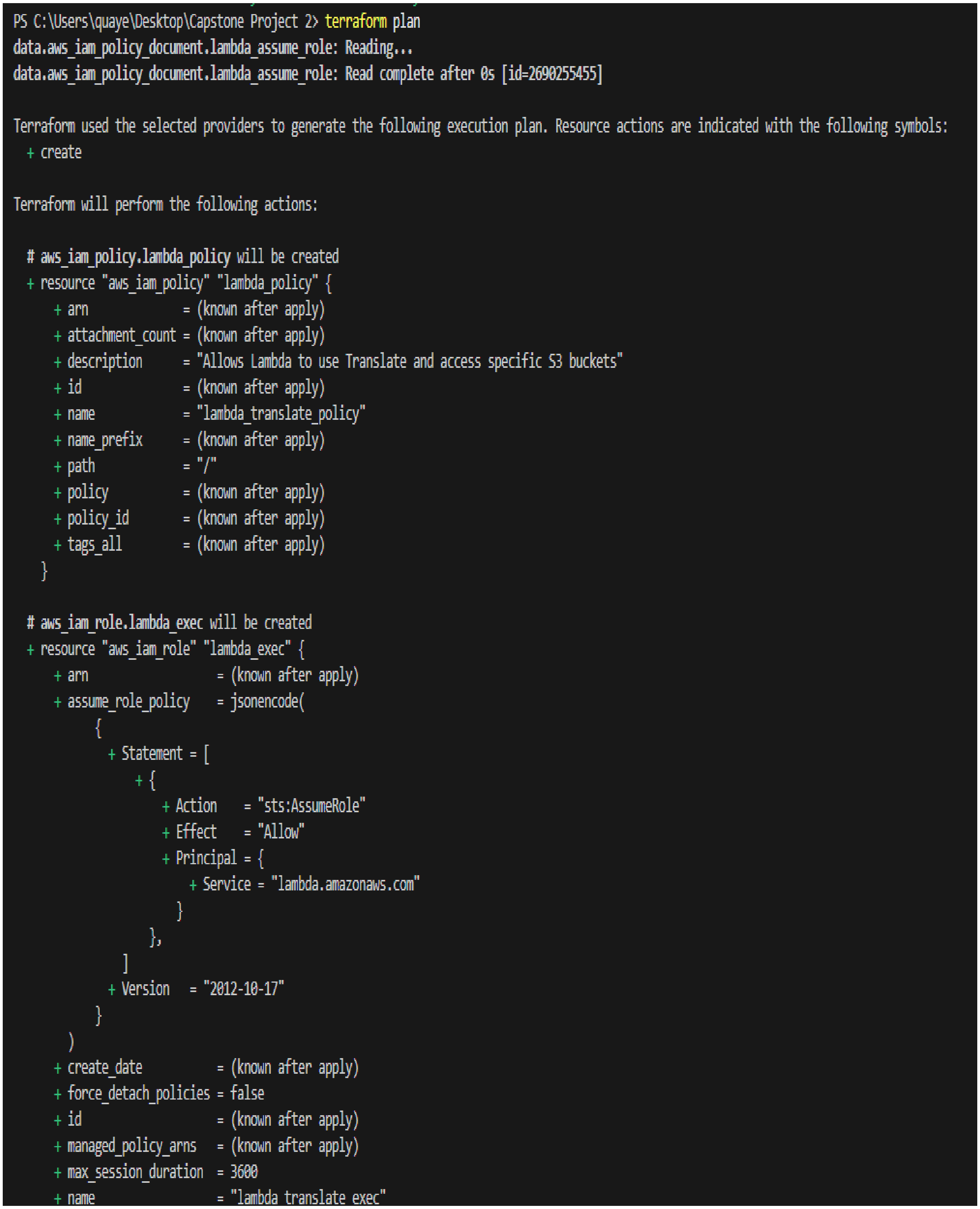
The terraform init command was successfully executed, setting up the working directory and installing required provider plugins.



```
PS C:\Users\quaye\Desktop\Capstone Project 2> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.11.0...
- Installed hashicorp/aws v6.11.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Terraform Planning

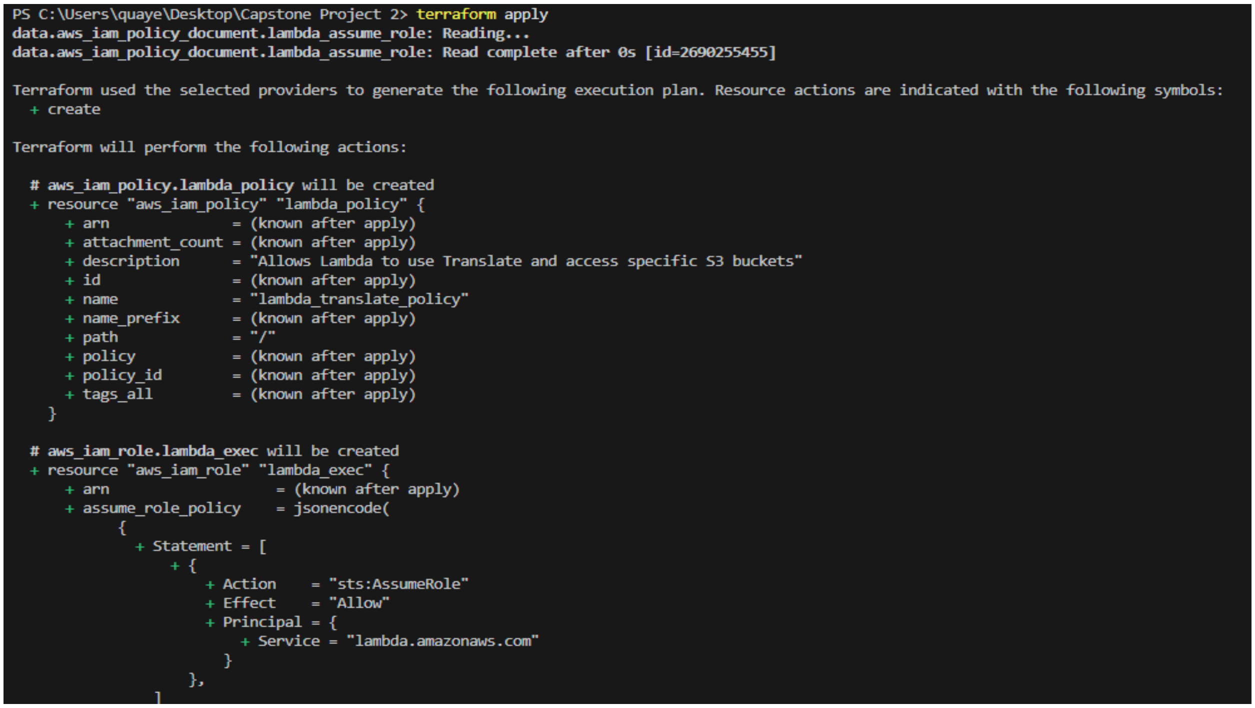
Created an execution plan to preview the changes Terraform would make:



```
PS C:\Users\quaye\Desktop\Capstone Project 2> terraform plan
data.aws_iam_policy_document.lambda_assume_role: Reading...
data.aws_iam_policy_document.lambda_assume_role: Read complete after 0s [id=2690255455]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_iam_policy.lambda_policy will be created
  + resource "aws_iam_policy" "lambda_policy" {
      + arn              = (known after apply)
      + attachment_count = (known after apply)
      + description      = "Allows Lambda to use Translate and access specific S3 buckets"
      + id               = (known after apply)
      + name             = "lambda_translate_policy"
      + name_prefix      = (known after apply)
      + path             = "/"
      + policy           = (known after apply)
      + policy_id        = (known after apply)
      + tags_all         = (known after apply)
    }

  # aws_iam_role.lambda_exec will be created
  + resource "aws_iam_role" "lambda_exec" {
      + arn                   = (known after apply)
      + assume_role_policy    = jsonencode(
            {
              + Statement = [
                  + {
                      + Action    = "sts:AssumeRole"
                      + Effect    = "Allow"
                      + Principal = {
                          + Service = "lambda.amazonaws.com"
                        }
                    },
                ]
              + Version   = "2012-10-17"
            }
        )
      + create_date           = (known after apply)
      + force_detach_policies = false
      + id                    = (known after apply)
      + managed_policy_arns   = (known after apply)
      + max_session_duration  = 3600
      + name                  = "lambda_translate_exec"
```

Terraform Apply

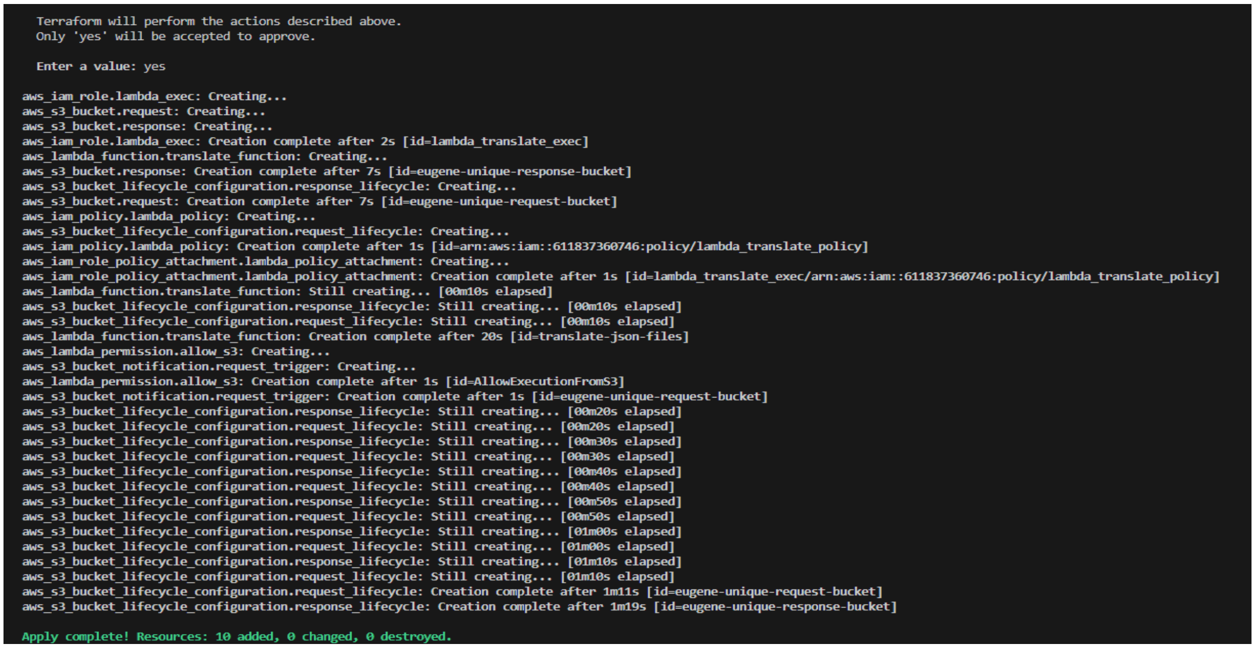
Applied the Terraform configuration to provision AWS resources. Confirmed the action by entering yes when prompted:



```
PS C:\Users\quaye\Desktop\Capstone Project 2> terraform apply
data.aws_iam_policy_document.lambda_assume_role: Reading...
data.aws_iam_policy_document.lambda_assume_role: Read complete after 0s [id=2690255455]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_iam_policy.lambda_policy will be created
  + resource "aws_iam_policy" "lambda_policy" {
      + arn              = (known after apply)
      + attachment_count = (known after apply)
      + description      = "Allows Lambda to use Translate and access specific S3 buckets"
      + id               = (known after apply)
      + name             = "lambda_translate_policy"
      + name_prefix      = (known after apply)
      + path             = "/"
      + policy           = (known after apply)
      + policy_id        = (known after apply)
      + tags_all         = (known after apply)
    }

  # aws_iam_role.lambda_exec will be created
  + resource "aws_iam_role" "lambda_exec" {
      + arn                   = (known after apply)
      + assume_role_policy    = jsonencode(
            {
              + Statement = [
                  + {
                      + Action    = "sts:AssumeRole"
                      + Effect    = "Allow"
                      + Principal = {
                          + Service = "lambda.amazonaws.com"
                        }
                    },
                ]
```

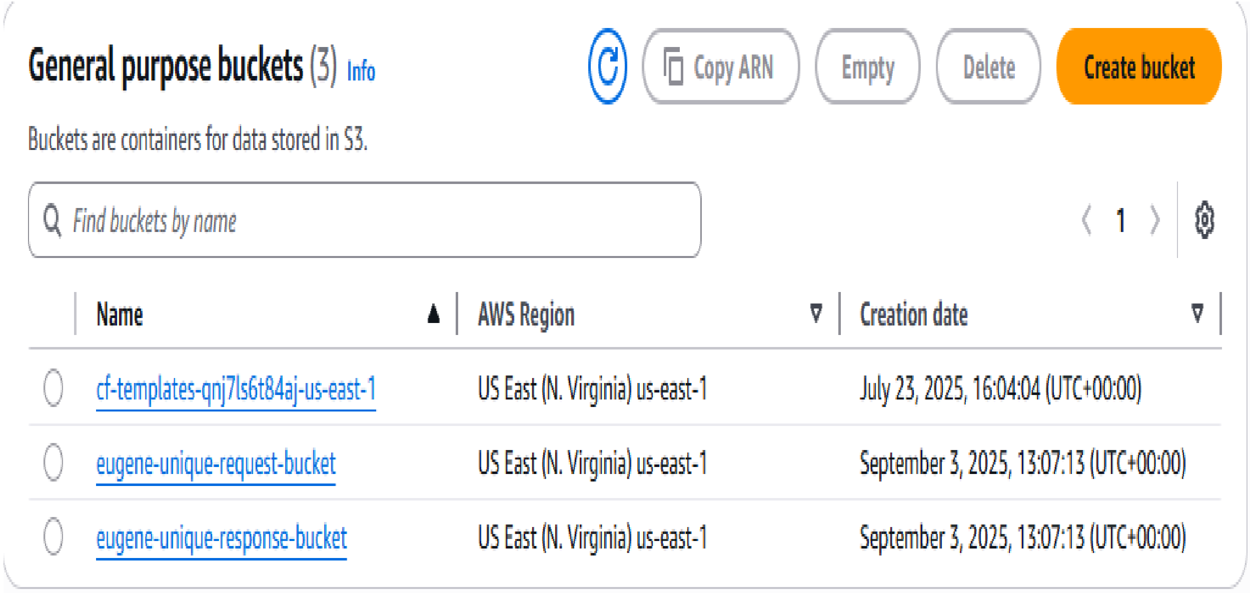
Resource Creation Confirmation

AWS resources were successfully created as defined in the Terraform configuration.



```
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_iam_role.lambda_exec: Creating...
aws_s3_bucket.request: Creating...
aws_s3_bucket.response: Creating...
aws_iam_role.lambda_exec: Creation complete after 2s [id=lambda_translate_exec]
aws_lambda_function.translate_function: Creating...
aws_s3_bucket.response: Creation complete after 7s [id=eugene-unique-response-bucket]
aws_s3_bucket_lifecycle_configuration.response_lifecycle: Creating...
aws_s3_bucket.request: Creation complete after 7s [id=eugene-unique-request-bucket]
aws_iam_policy.lambda_policy: Creating...
aws_s3_bucket_lifecycle_configuration.request_lifecycle: Creating...
aws_iam_policy.lambda_policy: Creation complete after 1s [id=arn:aws:iam::611837360746:policy/lambda_translate_policy]
aws_iam_role_policy_attachment.lambda_policy_attachment: Creating...
aws_iam_role_policy_attachment.lambda_policy_attachment: Creation complete after 1s [id=lambda_translate_exec/arn:aws:iam::611837360746:policy/lambda_translate_policy]
aws_lambda_function.translate_function: Still creating... [00m10s elapsed]
aws_s3_bucket_lifecycle_configuration.response_lifecycle: Still creating... [00m10s elapsed]
aws_s3_bucket_lifecycle_configuration.request_lifecycle: Still creating... [00m10s elapsed]
aws_lambda_function.translate_function: Creation complete after 20s [id=translate-json-files]
aws_lambda_permission.allow_s3: Creating...
aws_s3_bucket_notification.request_trigger: Creating...
aws_lambda_permission.allow_s3: Creation complete after 1s [id=AllowExecutionFromS3]
aws_s3_bucket_notification.request_trigger: Creation complete after 1s [id=eugene-unique-request-bucket]
aws_s3_bucket_lifecycle_configuration.response_lifecycle: Still creating... [00m20s elapsed]
aws_s3_bucket_lifecycle_configuration.request_lifecycle: Still creating... [00m20s elapsed]
aws_s3_bucket_lifecycle_configuration.response_lifecycle: Still creating... [00m30s elapsed]
aws_s3_bucket_lifecycle_configuration.request_lifecycle: Still creating... [00m30s elapsed]
aws_s3_bucket_lifecycle_configuration.response_lifecycle: Still creating... [00m40s elapsed]
aws_s3_bucket_lifecycle_configuration.request_lifecycle: Still creating... [00m40s elapsed]
aws_s3_bucket_lifecycle_configuration.response_lifecycle: Still creating... [00m50s elapsed]
aws_s3_bucket_lifecycle_configuration.request_lifecycle: Still creating... [00m50s elapsed]
aws_s3_bucket_lifecycle_configuration.response_lifecycle: Still creating... [01m00s elapsed]
aws_s3_bucket_lifecycle_configuration.request_lifecycle: Still creating... [01m00s elapsed]
aws_s3_bucket_lifecycle_configuration.response_lifecycle: Still creating... [01m10s elapsed]
aws_s3_bucket_lifecycle_configuration.request_lifecycle: Still creating... [01m10s elapsed]
aws_s3_bucket_lifecycle_configuration.request_lifecycle: Creation complete after 1m11s [id=eugene-unique-request-bucket]
aws_s3_bucket_lifecycle_configuration.response_lifecycle: Creation complete after 1m19s [id=eugene-unique-response-bucket]

Apply complete! Resources: 10 added, 0 changed, 0 destroyed.
```

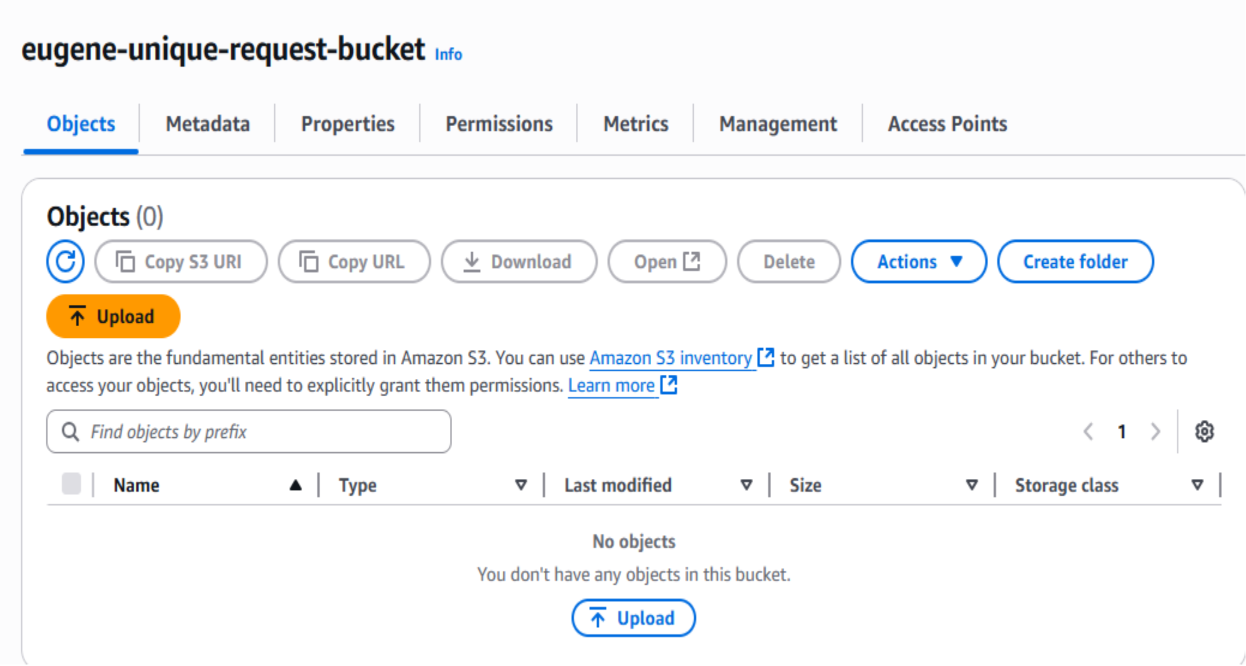
The screenshot confirms successful creation of the required S3 buckets during Terraform deployment.



Buckets are containers for data stored in S3.

	Name	AWS Region	Creation date
○	cf-templates-qnj7ls6t84aj-us-east-1	US East (N. Virginia) us-east-1	July 23, 2025, 16:04:04 (UTC+00:00)
○	eugene-unique-request-bucket	US East (N. Virginia) us-east-1	September 3, 2025, 13:07:13 (UTC+00:00)
○	eugene-unique-response-bucket	US East (N. Virginia) us-east-1	September 3, 2025, 13:07:13 (UTC+00:00)

The eugene-unique-request-bucket has been successfully created. No objects have been uploaded yet.



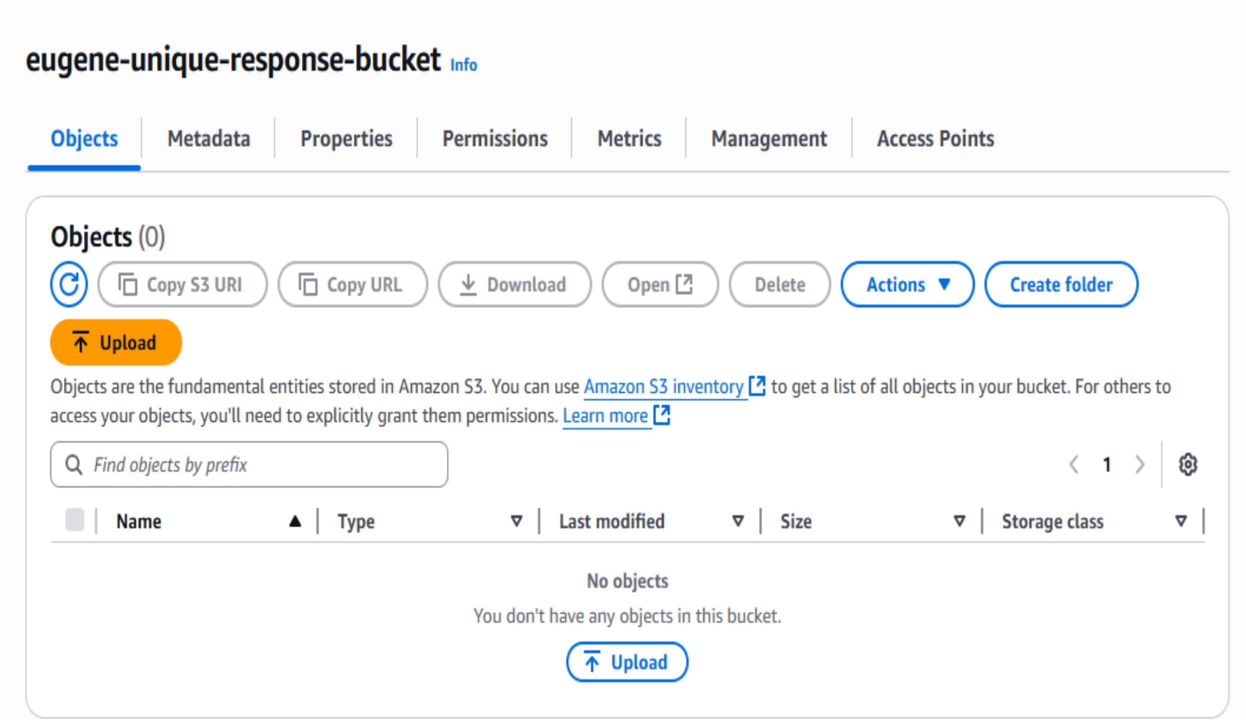
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class

No objects

You don't have any objects in this bucket.

The eugene-unique-response-bucket has been successfully created and is ready for use



eugene-unique-response-bucket

Objects (0)

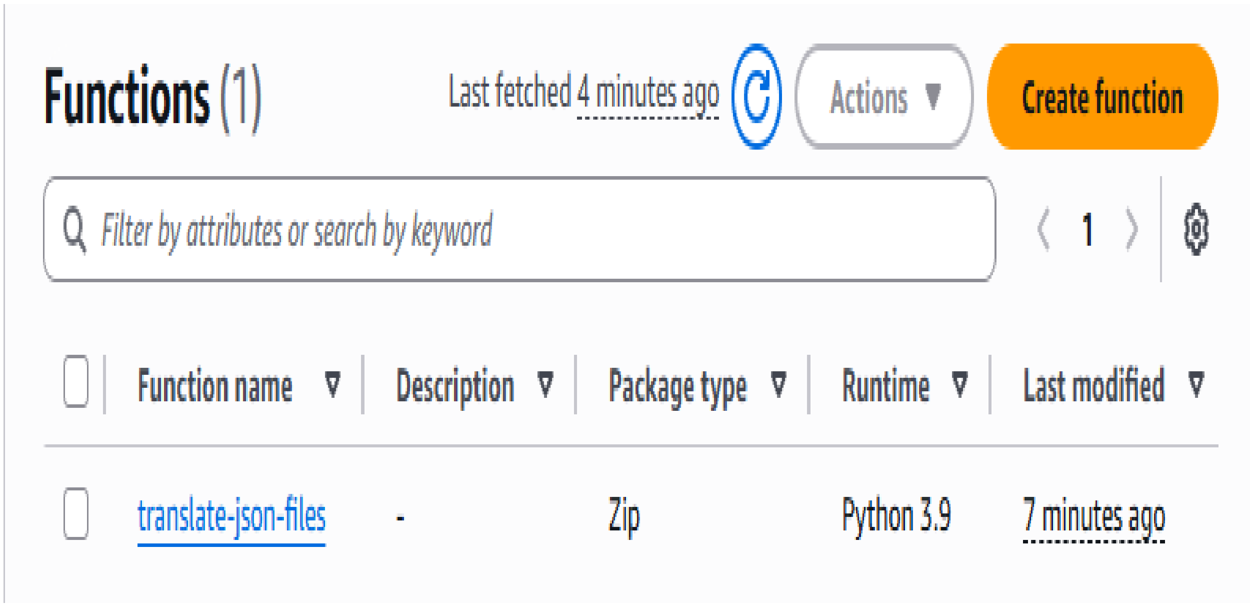
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type ▽	Last modified ▽	Size ▽	Storage class ▽

No objects

You don't have any objects in this bucket.

The translate-json-files Lambda function has been successfully deployed using a ZIP package with Python 3.9 runtime.

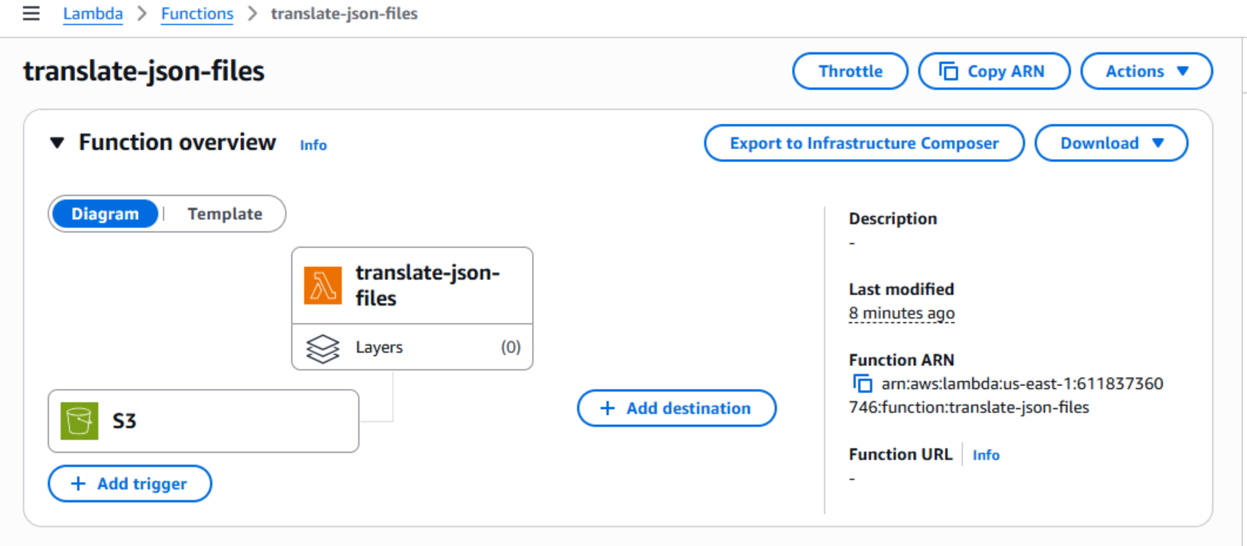


Functions (1)

Last fetched 4 minutes ago

□	Function name ▽	Description ▽	Package type ▽	Runtime ▽	Last modified ▽
□	translate-json-files	-	Zip	Python 3.9	7 minutes ago

The translate-json-files Lambda function is successfully configured with an S3 bucket trigger.



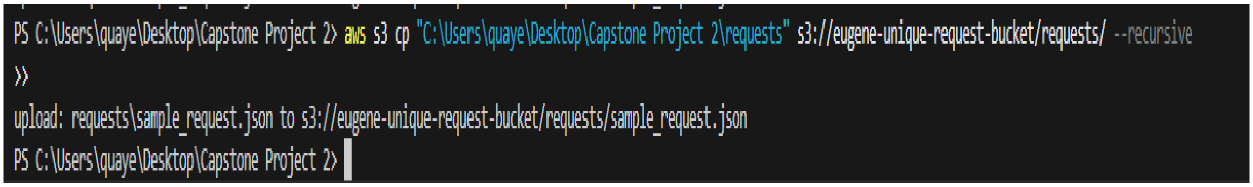
Description
-

Last modified
8 minutes ago

Function ARN
arn:aws:lambda:us-east-1:611837360746:function:translate-json-files

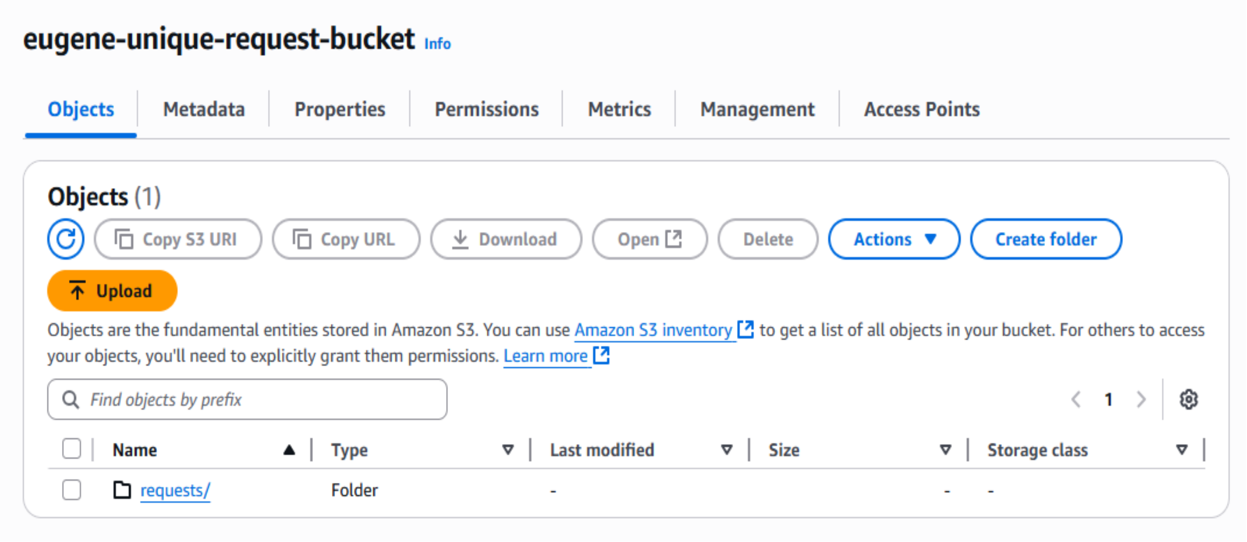
Function URL Info
-

Confirmed upload of sample_request.json to the eugene-unique-request-bucket via AWS CLI.



```
PS C:\Users\quaye\Desktop\Capstone Project 2> aws s3 cp "C:\Users\quaye\Desktop\Capstone Project 2\requests" s3://eugene-unique-request-bucket/requests/ --recursive
>>
upload: requests\sample_request.json to s3://eugene-unique-request-bucket/requests/sample_request.json
PS C:\Users\quaye\Desktop\Capstone Project 2>
```

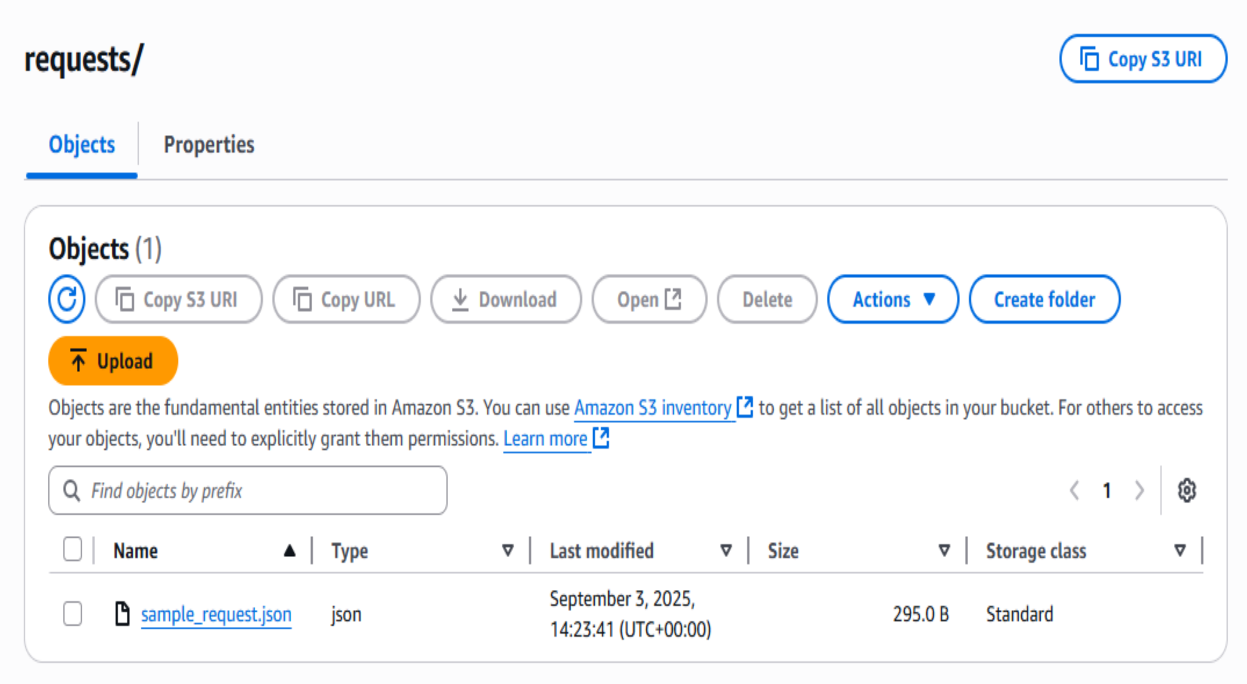
The requests/ folder has been successfully created in the eugene-unique-request-bucket.



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	requests/	Folder	-	-	-

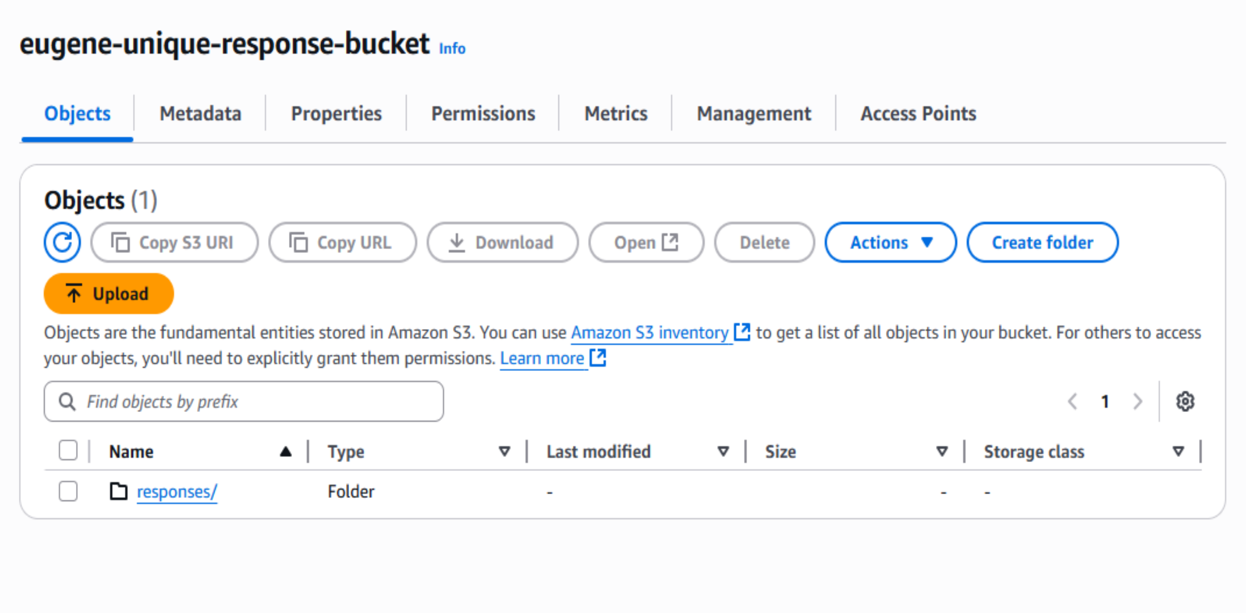
The sample_request.json file is successfully stored in the requests/ folder of the eugene-unique-request-bucket.



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type	Last modified	Size	Storage class
□	sample_request.json	json	September 3, 2025, 14:23:41 (UTC+00:00)	295.0 B	Standard

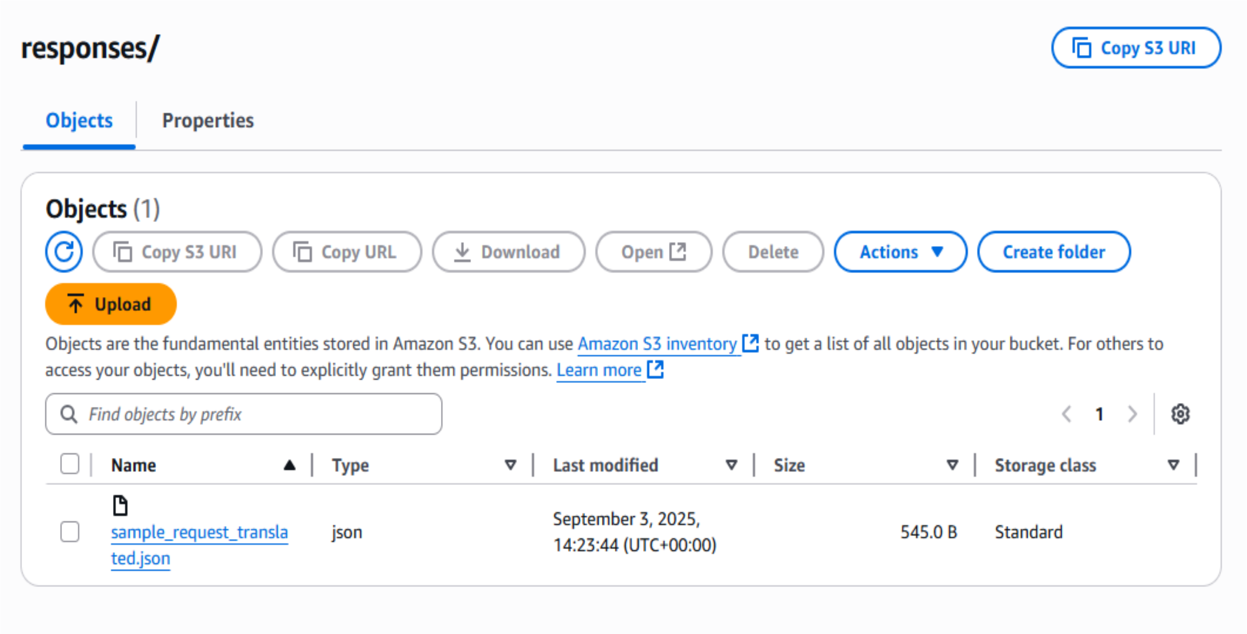
The responses/ folder has been successfully created in the eugene-unique-response-bucket.



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type	Last modified	Size	Storage class
□	responses/	Folder	-	-	-

The translated file sample_request_translated.json has been successfully stored in the responses/ folder of the eugene-unique-response-bucket.



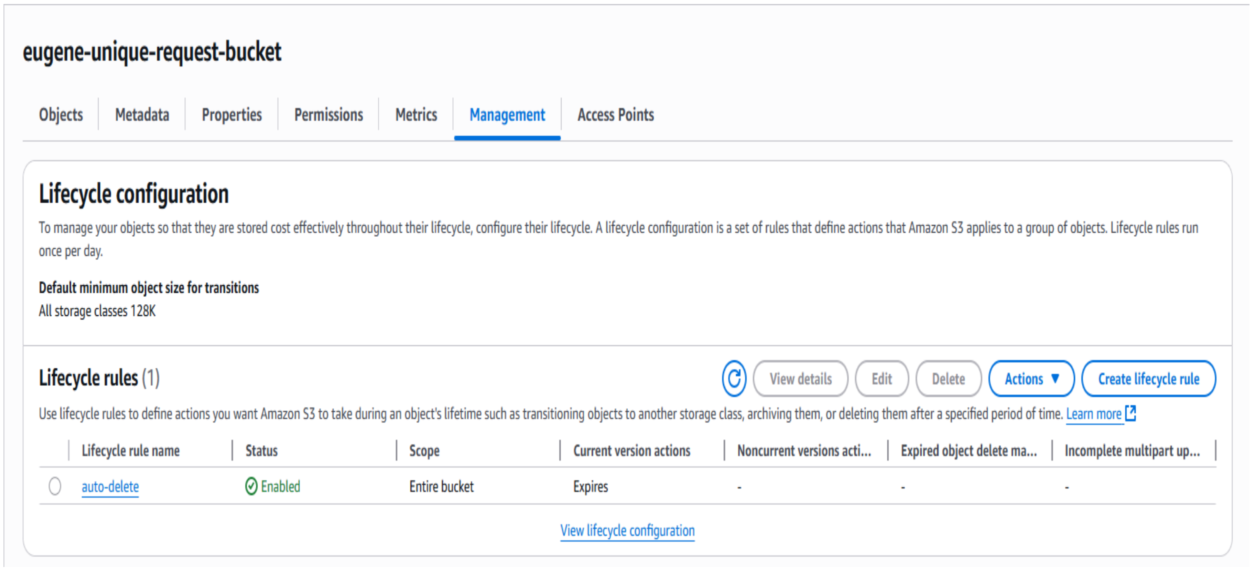
responses/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name ▲	Type	Last modified	Size	Storage class
□	sample_request_translated.json	json	September 3, 2025, 14:23:44 (UTC+00:00)	545.0 B	Standard

An auto-delete lifecycle rule has been successfully enabled for the eugene-unique-request-bucket.



eugene-unique-request-bucket

Lifecycle configuration

To manage your objects so that they are stored cost effectively throughout their lifecycle, configure their lifecycle. A lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. Lifecycle rules run once per day.

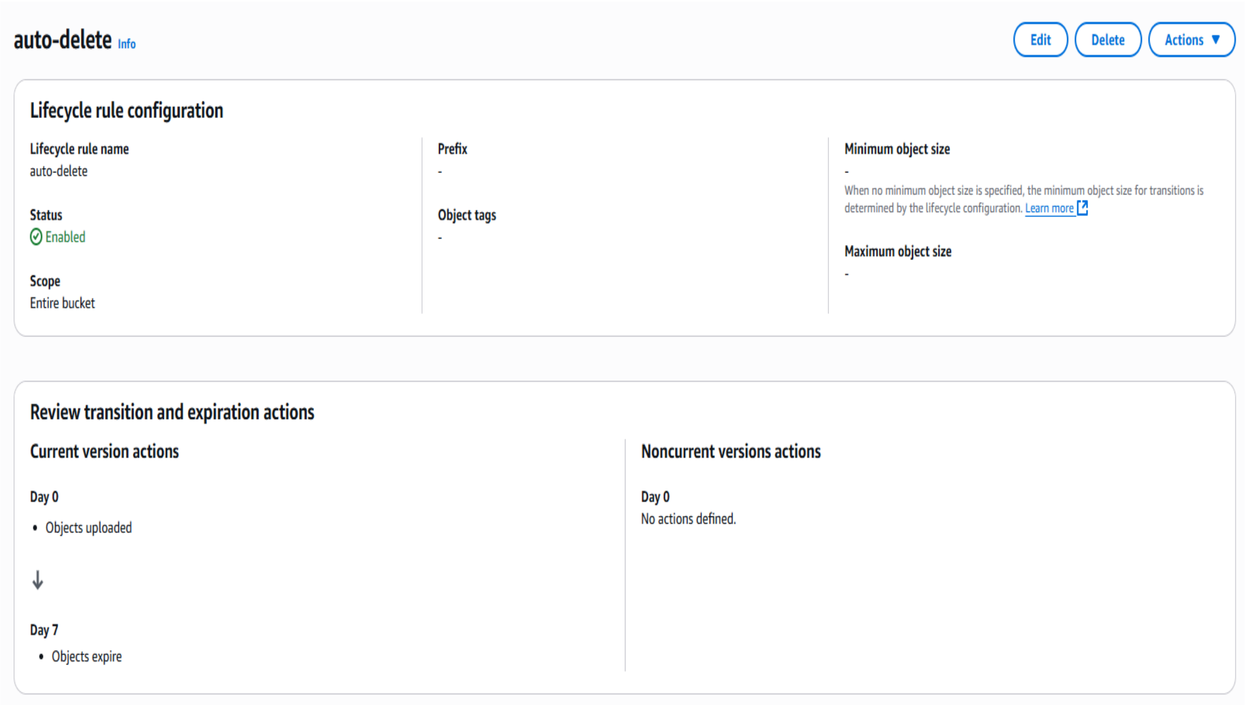
Default minimum object size for transitions
All storage classes 128K

Lifecycle rules (1)

Use lifecycle rules to define actions you want Amazon S3 to take during an object's lifetime such as transitioning objects to another storage class, archiving them, or deleting them after a specified period of time. Learn more

	Lifecycle rule name	Status	Scope	Current version actions	Noncurrent versions acti...	Expired object delete ma...	Incomplete multipart up...
○	auto-delete	✓ Enabled	Entire bucket	Expires	-	-	-

The auto-delete lifecycle rule is configured to expire objects in the bucket 7 days after upload.



auto-delete

Lifecycle rule configuration

Lifecycle rule name
auto-delete

Status
✓ Enabled

Scope
Entire bucket

Prefix
-

Object tags
-

Minimum object size
-
When no minimum object size is specified, the minimum object size for transitions is determined by the lifecycle configuration. Learn more

Maximum object size
-

Review transition and expiration actions

Current version actions

Day 0
- Objects uploaded

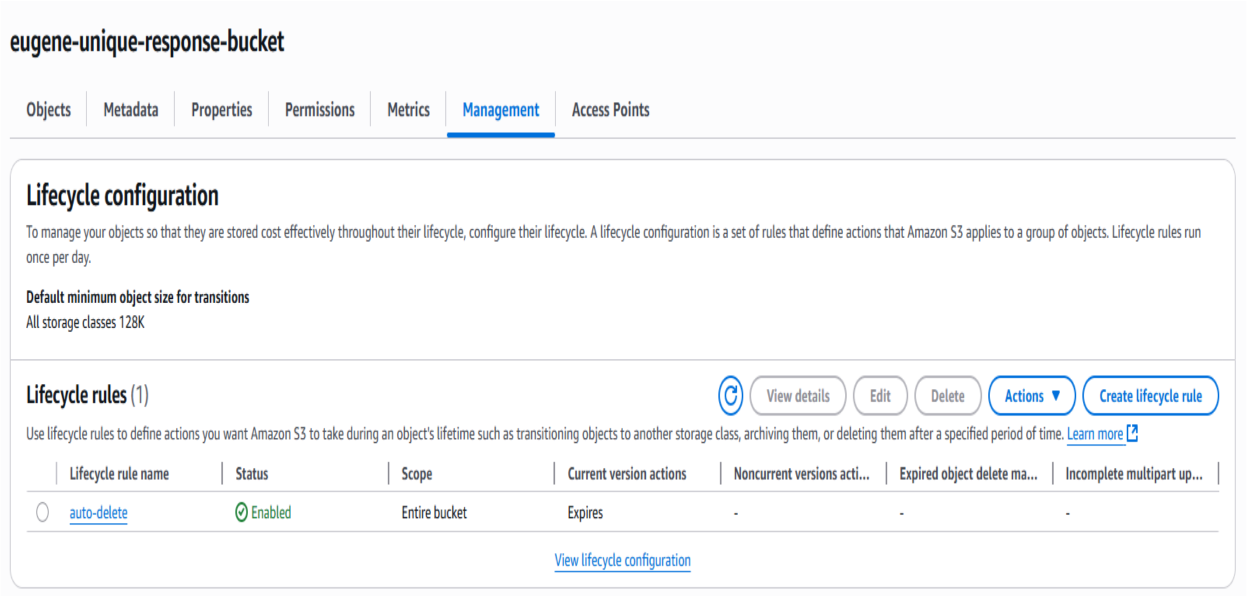
↓

Day 7
- Objects expire

Noncurrent versions actions

Day 0
No actions defined.

An auto-delete lifecycle rule is active for the eugene-unique-response-bucket, set to expire current version objects.



eugene-unique-response-bucket

Lifecycle configuration

To manage your objects so that they are stored cost effectively throughout their lifecycle, configure their lifecycle. A lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. Lifecycle rules run once per day.

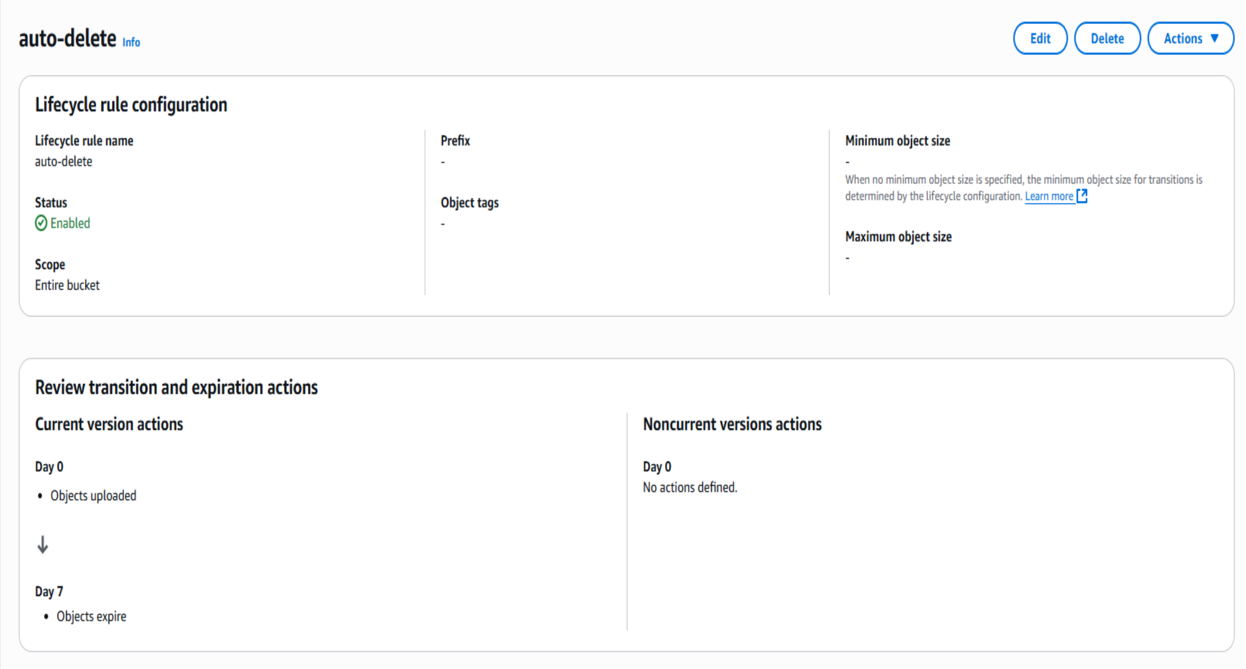
Default minimum object size for transitions
All storage classes 128K

Lifecycle rules (1)

Use lifecycle rules to define actions you want Amazon S3 to take during an object's lifetime such as transitioning objects to another storage class, archiving them, or deleting them after a specified period of time. Learn more

	Lifecycle rule name	Status	Scope	Current version actions	Noncurrent versions acti...	Expired object delete ma...	Incomplete multipart up...
○	auto-delete	✓ Enabled	Entire bucket	Expires	-	-	-

The auto-delete lifecycle rule for the response bucket is set to expire objects 7 days after upload.



Lifecycle rule configuration

Lifecycle rule name
auto-delete

Status
✓ Enabled

Scope
Entire bucket

Prefix
-

Object tags
-

Minimum object size
-
When no minimum object size is specified, the minimum object size for transitions is determined by the lifecycle configuration. Learn more

Maximum object size
-

Review transition and expiration actions

Current version actions

Day 0
- Objects uploaded

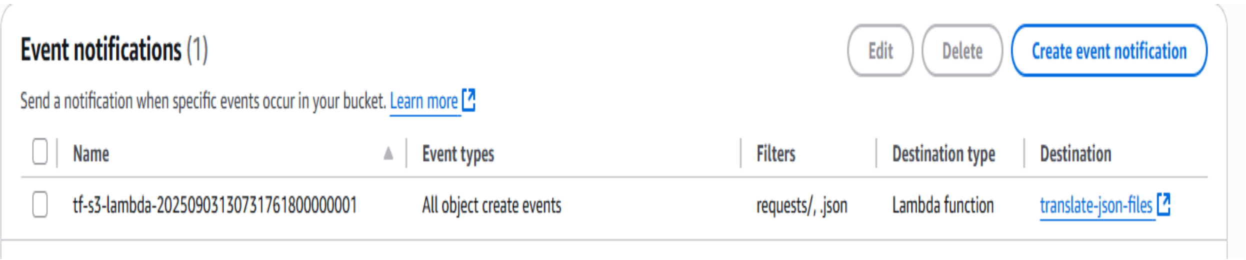
↓

Day 7
- Objects expire

Noncurrent versions actions

Day 0
No actions defined.

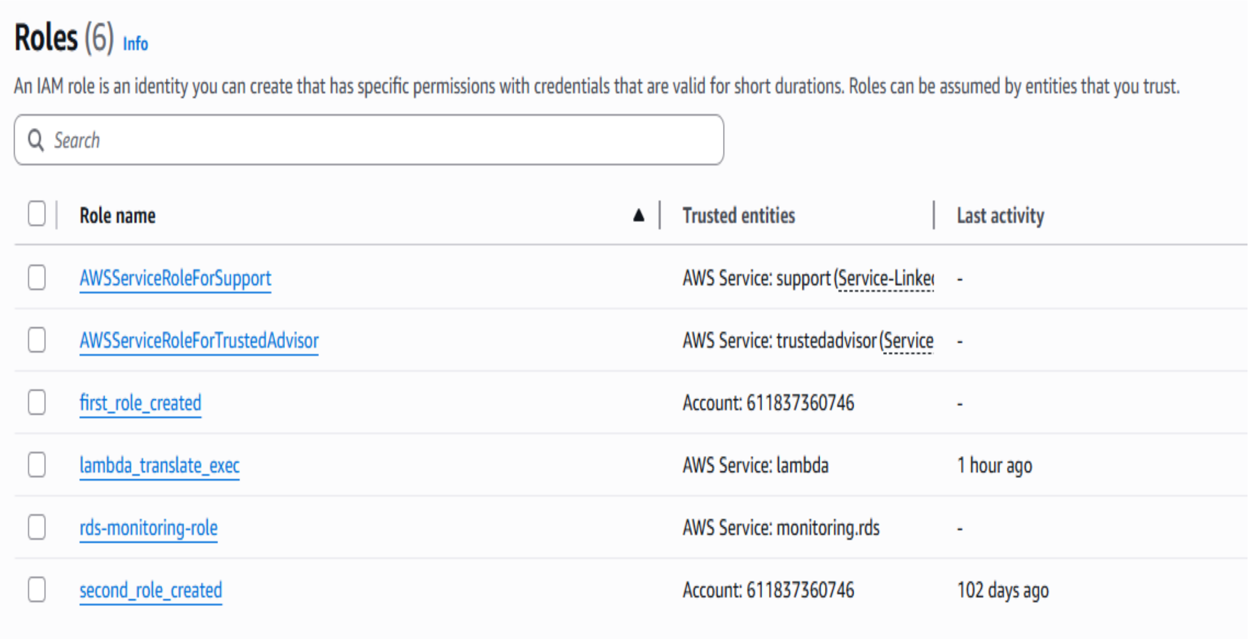
An event notification has been set up to trigger the translate-json-files Lambda function on all object creation events in the requests/ folder.



Send a notification when specific events occur in your bucket. Learn more

□ Name	Event types	Filters	Destination type	Destination
□ tf-s3-lambda-20250903130731761800000001	All object create events	requests/, .json	Lambda function	translate-json-files

The lambda_translate_exec role has been actively used by the Lambda function, confirming successful role assignment.

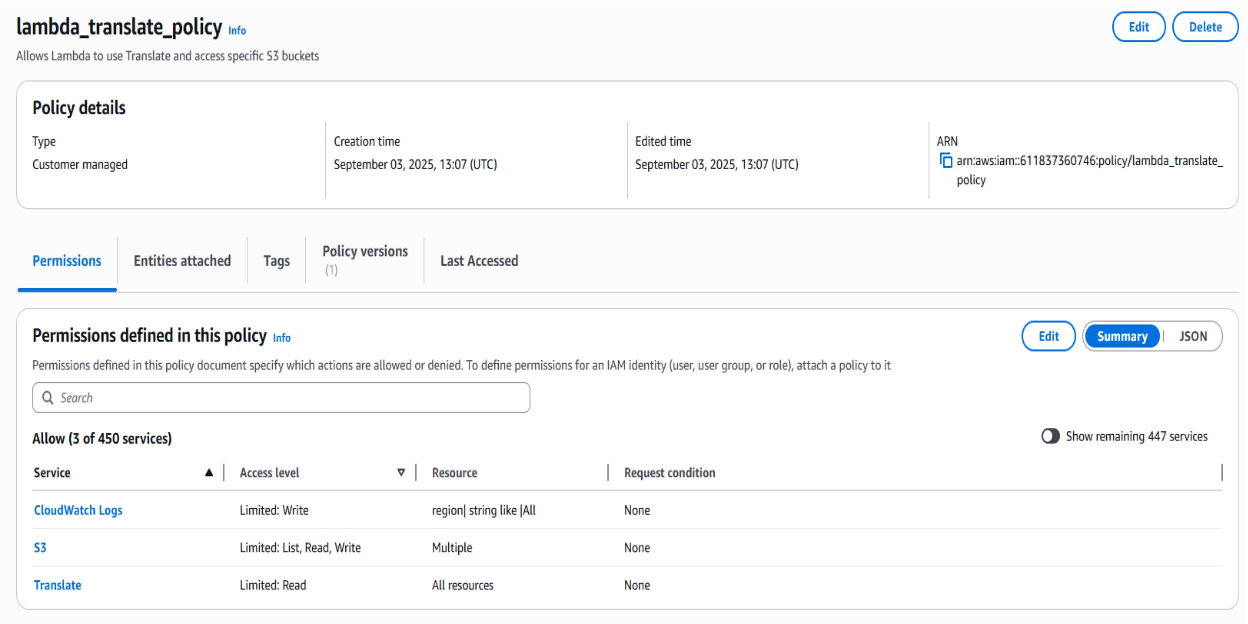


Roles (6) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

□	Role name ▲	Trusted entities	Last activity
□	AWSServiceRoleForSupport	AWS Service: support (Service-Linke	-
□	AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-
□	first_role_created	Account: 611837360746	-
□	lambda_translate_exec	AWS Service: lambda	1 hour ago
□	rds-monitoring-role	AWS Service: monitoring.rds	-
□	second_role_created	Account: 611837360746	102 days ago

The lambda_translate_policy grants necessary permissions for Lambda to access Amazon Translate and interact with designated S3 buckets.



lambda_translate_policy Info

Allows Lambda to use Translate and access specific S3 buckets

Policy details

Type	Creation time	Edited time	ARN
Customer managed	September 03, 2025, 13:07 (UTC)	September 03, 2025, 13:07 (UTC)	arn:aws:iam::611837360746:policy/lambda_translate_policy

Permissions defined in this policy Info

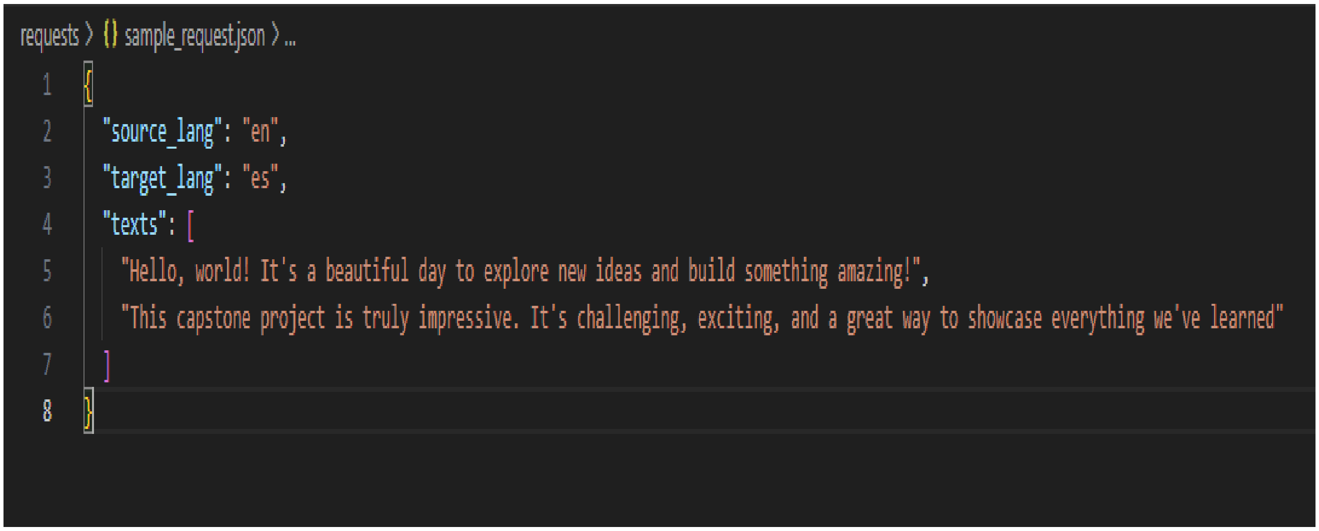
Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Allow (3 of 450 services)

Service ▲	Access level ▽	Resource	Request condition
CloudWatch Logs	Limited: Write	region| string like |All	None
S3	Limited: List, Read, Write	Multiple	None
Translate	Limited: Read	All resources	None

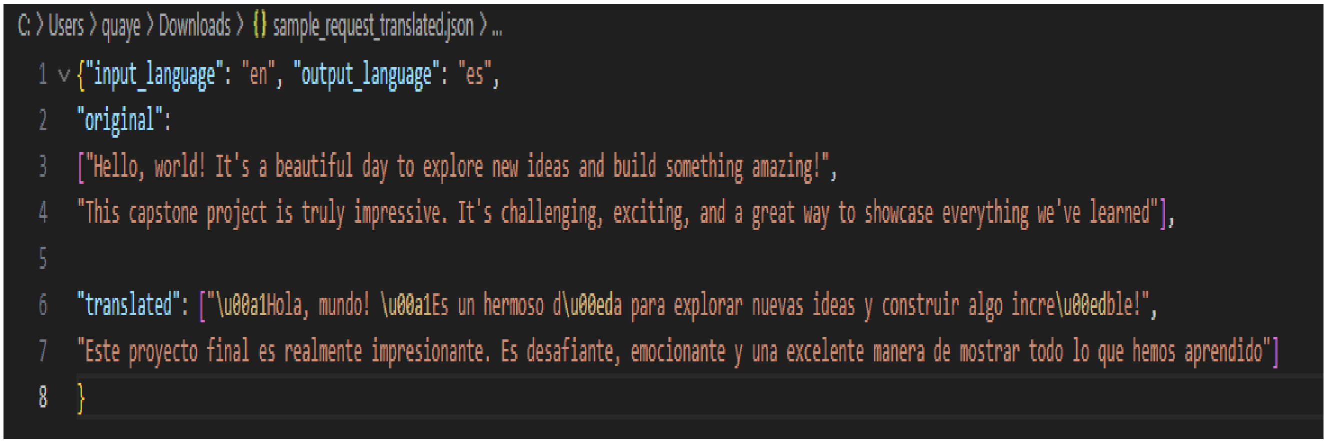
Testing & Validation

Sample Input **(requests/sample_request.json):**



```
{
  "source_lang": "en",
  "target_lang": "es",
  "texts": [
    "Hello, world! It's a beautiful day to explore new ideas and build something amazing!",
    "This capstone project is truly impressive. It's challenging, exciting, and a great way to showcase everything we've learned"
  ]
}
```

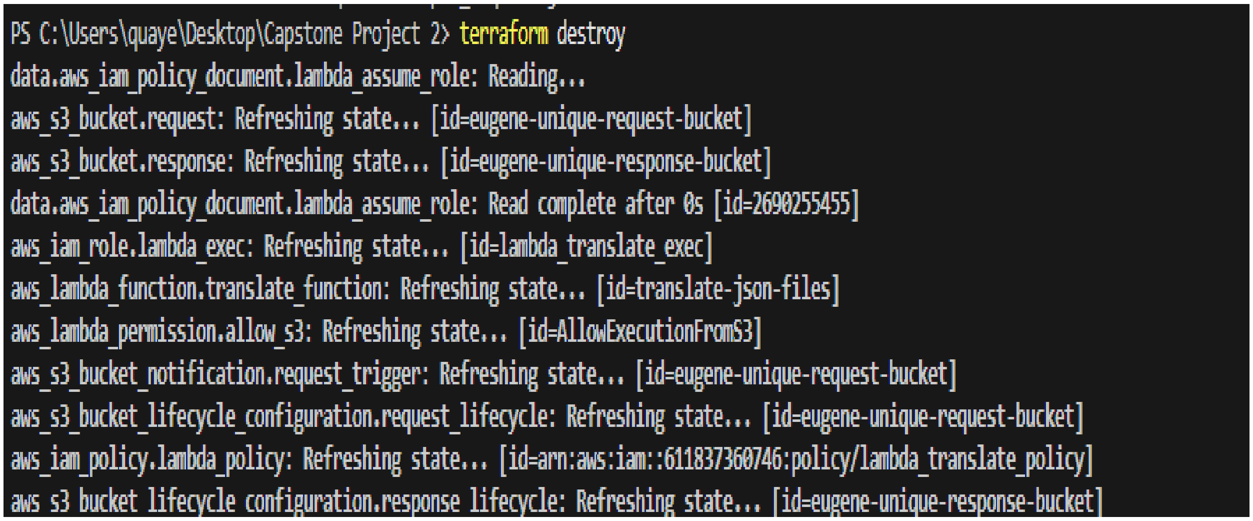
Expected Output **(responses/sample_request_translated.json):**


C: > Users > quaye > Downloads > {} sample_request_translated.json > ...

```
{"input_language": "en", "output_language": "es",
"original":
["Hello, world! It's a beautiful day to explore new ideas and build something amazing!",
"This capstone project is truly impressive. It's challenging, exciting, and a great way to showcase everything we've learned"],

"translated": ["\u00a1Hola, mundo! \u00a1Es un hermoso d\u00eda para explorar nuevas ideas y construir algo incre\u00edble!",
"Este proyecto final es realmente impresionante. Es desafiante, emocionante y una excelente manera de mostrar todo lo que hemos aprendido"]
}
```

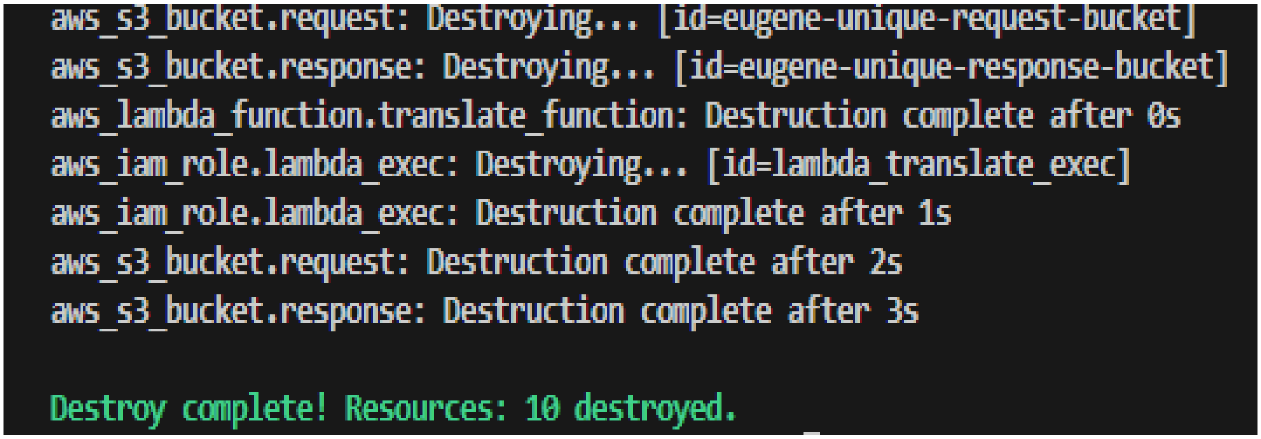
- Use CLI to upload input files.
- Check Lambda logs in CloudWatch for errors.
- Download response files for validation.

The terraform destroy command was executed, beginning the teardown of all previously provisioned AWS resources.



```
PS C:\Users\quaye\Desktop\Capstone Project 2> terraform destroy
data.aws_iam_policy_document.lambda_assume_role: Reading...
aws_s3_bucket.request: Refreshing state... [id=eugene-unique-request-bucket]
aws_s3_bucket.response: Refreshing state... [id=eugene-unique-response-bucket]
data.aws_iam_policy_document.lambda_assume_role: Read complete after 0s [id=2690255455]
aws_iam_role.lambda_exec: Refreshing state... [id=lambda_translate_exec]
aws_lambda_function.translate_function: Refreshing state... [id=translate-json-files]
aws_lambda_permission.allow_s3: Refreshing state... [id=AllowExecutionFromS3]
aws_s3_bucket_notification.request_trigger: Refreshing state... [id=eugene-unique-request-bucket]
aws_s3_bucket_lifecycle_configuration.request_lifecycle: Refreshing state... [id=eugene-unique-request-bucket]
aws_iam_policy.lambda_policy: Refreshing state... [id=arn:aws:iam::611837360746:policy/lambda_translate_policy]
aws_s3_bucket_lifecycle_configuration.response_lifecycle: Refreshing state... [id=eugene-unique-response-bucket]
```

All provisioned AWS resources, including S3 buckets, Lambda function, and IAM roles, have been successfully destroyed.



```
aws_s3_bucket.request: Destroying... [id=eugene-unique-request-bucket]
aws_s3_bucket.response: Destroying... [id=eugene-unique-response-bucket]
aws_lambda_function.translate_function: Destruction complete after 0s
aws_iam_role.lambda_exec: Destroying... [id=lambda_translate_exec]
aws_iam_role.lambda_exec: Destruction complete after 1s
aws_s3_bucket.request: Destruction complete after 2s
aws_s3_bucket.response: Destruction complete after 3s

Destroy complete! Resources: 10 destroyed.
```

Challenges

During the development of this serverless AWS NLP Translation Pipeline, several challenges were encountered which provided valuable learning experiences:

- **Lambda Deployment Packaging:** Packaging and uploading the Lambda function as a ZIP file was error-prone, especially regarding path references in Terraform and keeping the deployed code in sync with the latest changes.
- **S3 Bucket Name Collisions:** AWS requires S3 bucket names to be globally unique. During testing, bucket creation sometimes failed due to name collisions, requiring careful planning and dynamic naming strategies.
- **Managing Free Tier Usage:** Close monitoring was needed to avoid exceeding AWS Free Tier quotas, especially during repeated testing, to prevent unexpected charges.

Lessons learned

This project led to deep insights relevant to both cloud engineering and DevOps best practices:

- **Infrastructure as Code (IaC) is Essential:** Using Terraform brought tremendous value in terms of repeatability, version control, and transparency, enabling fast re-provisioning of environments and straightforward rollback of changes.
- **Security First:** Least privilege IAM policy design is not just best practice, but crucial for real-world deployments. Early investment in security policy design saved significant troubleshooting effort later.
- **Observability is Key:** Effective use of AWS CloudWatch logs proved to be essential for debugging distributed serverless architectures. Proper instrumentation and logging should be included from the start of all cloud projects.
- **Automation Saves Time:** Automating packaging and Lambda deployment processes eliminated many manual errors and ensured continuous delivery of the most up-to-date application code.
- **Resource Naming and State Management:** The globally unique naming requirement for S3 and state management in IaC highlighted the importance of careful resource planning in cloud projects.

Cost management

Throughout the project, careful attention was paid to control costs and remain within the AWS Free Tier:

- **Lifecycle Policies:** Implemented on all S3 buckets to automatically expire and delete data after seven days, minimizing storage costs and preventing orphaned resources.
- **Free Tier Resource Usage:** Selected only services included in AWS Free Tier (Lambda, S3, Translate) and kept overall usage well below monthly limits by using small sample files and limiting translation operations.
- **Minimal Permissions:** Used narrowly scoped IAM policies and roles, reducing the risk of accidental access to costly AWS services outside of the project's requirements.
- **Regular Teardown:** Enforced a practice of destroying infrastructure (terraform destroy) immediately upon project completion or after test iterations, ensuring no lingering or idle resources run up costs.
- **Monitoring:** Utilized the AWS Billing Dashboard for continuous oversight of resource consumption and early detection of any potential overages.

Conclusion

This capstone project demonstrates the successful design and deployment of a secure, cost-effective, and scalable serverless NLP translation pipeline using AWS and Terraform. Leveraging infrastructure as code and managed cloud services enables extremely rapid iteration, high availability, and robust security without incurring the overhead of traditional infrastructure management.

The pipeline automates the process from resource provisioning, through translation workflows, to secure data storage, with full traceability and cost control. The experiences and skills gained through this project—including proficiency in Infrastructure as Code (IaC), serverless computing, cloud security, and cost-effective cloud engineering—provide a strong foundation for developing modern cloud-native solutions and contribute valuable preparation for future real-world deployments.

References

- AWS Translate: https://docs.aws.amazon.com/translate/latest/dg/what-is.html
- Terraform AWS Provider: https://registry.terraform.io/providers/hashicorp/aws/latest/docs
- AWS Lambda Python: https://docs.aws.amazon.com/lambda/latest/dg/python-handler.html
- AWS S3: https://docs.aws.amazon.com/s3/